

#6  
SD  
5/28/02

JPO1/408

PCT/JP 01/00408

日 本 国 特 許 庁

PATENT OFFICE  
JAPANESE GOVERNMENT

28.01.01

REC'D 09 MAR 2001

WIPO

PCT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日  
Date of Application:

2000年 7月14日

出 願 番 号  
Application Number:

特願2000-213988

出 願 人  
Applicant(s):

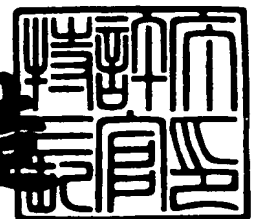
株式会社ナムコ

PRIORITY  
DOCUMENT  
SUBMITTED OR TRANSMITTED IN  
COMPLIANCE WITH RULE 17.1(a) OR (b)

2001年 2月23日

特許庁長官  
Commissioner,  
Patent Office

及川耕造



出証番号 出証特2001-3009688

【書類名】 特許願

【整理番号】 NM-0132401

---

【提出日】 平成12年 7月14日

【あて先】 特許庁長官殿

【国際特許分類】 A63F 9/22

【発明者】

【住所又は居所】 東京都大田区多摩川2丁目8番5号 株式会社ナムコ内

【氏名】 橘高 繁

【特許出願人】

【識別番号】 000134855

【氏名又は名称】 株式会社ナムコ

【代理人】

【識別番号】 100090387

【弁理士】

【氏名又は名称】 布施 行夫

【電話番号】 03-5397-0891

【選任した代理人】

【識別番号】 100090479

---

【弁理士】

【氏名又は名称】 井上 一

【電話番号】 03-5397-0891

【選任した代理人】

【識別番号】 100090398

【弁理士】

【氏名又は名称】 大淵 美千栄

【電話番号】 03-5397-0891

【先の出願に基づく優先権主張】

【出願番号】 特願2000- 20464

【出願日】 平成12年 1月28日

【手数料の表示】

【予納台帳番号】 039479

---

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9814051

【ブルーフの要否】 要

【書類名】 明細書

【発明の名称】 ゲームシステム及び情報記憶媒体

【特許請求の範囲】

【請求項1】 画像生成を行うゲームシステムであって、

元画像の各画素の奥行き値を、インデックスカラー・テクスチャマッピング用のルックアップテーブルのインデックス番号として設定する手段と、

元画像の各画素の奥行き値がインデックス番号として設定された前記ルックアップテーブルを用いて、仮想オブジェクトに対してインデックスカラー・テクスチャマッピングを行い、元画像の各画素の奥行き値に応じた値に各画素の $\alpha$ 値を設定する手段と、

元画像と該元画像に対応するぼかし画像とを、各画素に設定される $\alpha$ 値に基づいて合成する手段と、

を含むことを特徴とするゲームシステム。

【請求項2】 請求項1において、

元画像の各画素の奥行き値を、前記奥行き値の最上位ビットよりも下位のビットI～ビットJにより構成される第2の奥行き値に変換し、前記第2の奥行き値を、インデックスカラー・テクスチャマッピング用の前記ルックアップテーブルのインデックス番号として設定することを特徴とするゲームシステム。

【請求項3】 請求項2において、

前記奥行き値のビットI～ビットJ以外のビットの値に応じて、前記第2の奥行き値を所与の値にクランプすることを特徴とするゲームシステム。

【請求項4】 請求項2又は3において、

前記奥行き値を、インデックスカラー・テクスチャマッピング用のルックアップテーブルのインデックス番号として設定し、前記ルックアップテーブルを用いて仮想オブジェクトに対してインデックスカラー・テクスチャマッピングを行い、前記奥行き値を前記第2の奥行き値に変換することを特徴とするゲームシステム。

【請求項5】 請求項2乃至4のいずれかにおいて、

前記奥行き値のビットM～ビットNを、インデックスカラー・テクスチャマッピング用の第1のルックアップテーブルのインデックス番号として設定し、前記

---

第1のルックアップテーブルを用いて仮想オブジェクトに対してインデックスカラー・テクスチャマッピングを行い、前記奥行き値を第3の奥行き値に変換し、

前記奥行き値のビットK～ビットL ( $K \geq I \geq L > M \geq J \geq N$ ) を、インデックスカラー・テクスチャマッピング用の第2のルックアップテーブルのインデックス番号として設定し、前記第2のルックアップテーブルを用いて仮想オブジェクトに対してインデックスカラー・テクスチャマッピングを行い、前記奥行き値を第4の奥行き値に変換し、

前記第3、第4の奥行き値に基づいて前記第2の奥行き値を求めることを特徴とするゲームシステム。

【請求項6】 請求項1乃至5のいずれかにおいて、

元画像をテクスチャとして設定し、該テクスチャをテクセル補間方式で仮想オブジェクトにマッピングする際に仮想オブジェクトのテクスチャ座標をシフトさせ、元画像のぼかし画像を生成することを特徴とするゲームシステム。

【請求項7】 請求項1乃至6のいずれかにおいて、

前記仮想オブジェクトが、表示画面サイズのポリゴンであることを特徴とするゲームシステム。

---

【請求項8】 請求項1乃至6のいずれかにおいて、

前記仮想オブジェクトが、表示画面を分割したブロックのサイズのポリゴンであることを特徴とするゲームシステム。

【請求項9】 コンピュータが使用可能な情報記憶媒体であって、

元画像の各画素の奥行き値を、インデックスカラー・テクスチャマッピング用のルックアップテーブルのインデックス番号として設定する手段と、

元画像の各画素の奥行き値がインデックス番号として設定された前記ルックアップテーブルを用いて、仮想オブジェクトに対してインデックスカラー・テクスチャマッピングを行い、元画像の各画素の奥行き値に応じた値に各画素の $\alpha$ 値を設定する手段と、

元画像と該元画像に対応するぼかし画像とを、各画素に設定される $\alpha$ 値に基づ

いて合成する手段と、

を実行するためのプログラムを含むことを特徴とする情報記憶媒体。

---

【請求項 10】 請求項 9 において、

元画像の各画素の奥行き値を、前記奥行き値の最上位ビットよりも下位のビット I ～ビット J により構成される第 2 の奥行き値に変換し、前記第 2 の奥行き値を、インデックスカラー・テクスチャマッピング用の前記ルックアップテーブルのインデックス番号として設定することを特徴とする情報記憶媒体。

【請求項 11】 請求項 10 において、

前記奥行き値のビット I ～ビット J 以外のビットの値に応じて、前記第 2 の奥行き値を所与の値にクランプすることを特徴とする情報記憶媒体。

【請求項 12】 請求項 10 又は 11 において、

前記奥行き値を、インデックスカラー・テクスチャマッピング用のルックアップテーブルのインデックス番号として設定し、前記ルックアップテーブルを用いて仮想オブジェクトに対してインデックスカラー・テクスチャマッピングを行い、前記奥行き値を前記第 2 の奥行き値に変換することを特徴とする情報記憶媒体。

【請求項 13】 請求項 10 乃至 12 のいずれかにおいて、

前記奥行き値のビット M ～ビット N を、インデックスカラー・テクスチャマッピング用の第 1 のルックアップテーブルのインデックス番号として設定し、前記第 1 のルックアップテーブルを用いて仮想オブジェクトに対してインデックスカラー・テクスチャマッピングを行い、前記奥行き値を第 3 の奥行き値に変換し、

前記奥行き値のビット K ～ビット L ( $K \geq I \geq L > M \geq J \geq N$ ) を、インデックスカラー・テクスチャマッピング用の第 2 のルックアップテーブルのインデックス番号として設定し、前記第 2 のルックアップテーブルを用いて仮想オブジェクトに対してインデックスカラー・テクスチャマッピングを行い、前記奥行き値を第 4 の奥行き値に変換し、

前記第 3、第 4 の奥行き値に基づいて前記第 2 の奥行き値を求めることを特徴とする情報記憶媒体。

【請求項 14】 請求項 9 乃至 13 のいずれかにおいて、

元画像をテクスチャとして設定し、該テクスチャをテクセル補間方式で仮想オブジェクトにマッピングする際に仮想オブジェクトのテクスチャ座標をシフトさせ、元画像のほかし画像を生成することを特徴とする情報記憶媒体。

【請求項 1 5】 請求項 9 乃至 1 4 のいずれかにおいて、  
前記仮想オブジェクトが、表示画面サイズのポリゴンであることを特徴とする情報記憶媒体。

【請求項 1 6】 請求項 9 乃至 1 4 のいずれかにおいて、  
前記仮想オブジェクトが、表示画面を分割したブロックのサイズのポリゴンであることを特徴とする情報記憶媒体。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は、ゲームシステム及び情報記憶媒体に関する。

【0 0 0 2】

【背景技術及び発明が解決しようとする課題】

従来より、仮想的な 3 次元空間であるオブジェクト空間内の所与の視点から見える画像を生成するゲームシステムが知られており、いわゆる仮想現実を体験できるものとして人気が高い。レーシングゲームを楽しむことができるゲームシステムを例にとれば、プレーヤは、車（オブジェクト）を操作してオブジェクト空間内で走行させ、他のプレーヤやコンピュータが操作する車と競争することで 3 次元ゲームを楽しむ。

【0 0 0 3】

さて、従来のゲームシステムにより生成される画像は、人間の視界画像のように視点からの距離に応じてフォーカシングされた画像ではなかった。このため、画像内の全ての被写体にピントが合っているかのような表現になっていた。

【0 0 0 4】

しかしながら、至近距離から遠距離までの全ての被写体にピントが合っている画像は、日常生活では見るできない画像であるため、見た目に不自然さがあった。

## 【0005】

よりリアリティを追求するためには、視点とオブジェクトとの距離や視線方向などに応じてピントの度合いが調節された画像を生成することが望ましい。しかしながら、ゲーム空間内の個々のオブジェクトについて視点との距離等を計算し、各オブジェクト毎にぼやけ具合を演算することで、ぼやけた画像を生成すると、処理負荷が過大になる。

## 【0006】

リアルタイムに変化する視点に対応した画像を、制約されたハードウェア資源を用いて生成する必要があるゲームシステムにおいては、如何にして少ない処理負担で、現実世界の視界画像のようにフォーカシングされた画像を生成するかが重要な課題となる。

## 【0007】

本発明は、以上のような課題に鑑みてなされたものであり、その目的とするところは、よりリアルな画像を少ない処理負担で生成できるゲームシステム及び情報記憶媒体を提供することにある。より具体的には、現実世界の視界画像のようにフォーカシングされた画像を、少ない処理負担で生成できるゲームシステム及び情報記憶媒体を提供することにある。

## 【0008】

## 【課題を解決するための手段】

上記課題を解決するために、本発明は、画像生成を行うゲームシステムであって、元画像の各画素の奥行き値を、インデックスカラー・テクスチャマッピング用のルックアップテーブルのインデックス番号として設定する手段と、元画像の各画素の奥行き値がインデックス番号として設定された前記ルックアップテーブルを用いて、仮想オブジェクトに対してインデックスカラー・テクスチャマッピングを行い、元画像の各画素の奥行き値に応じた値に各画素の $\alpha$ 値を設定する手段と、元画像と該元画像に対応するぼかし画像とを、各画素に設定される $\alpha$ 値に基づいて合成する手段とを含むことを特徴とする。また本発明に係る情報記憶媒体は、コンピュータにより使用可能な情報記憶媒体であって、上記手段を実行するためのプログラムを含むことを特徴とする。また本発明に係るプログラムは、



コンピュータにより使用可能なプログラム（搬送波に具現化されるプログラムを含む）であって、上記手段を実行するための処理ルーチンを含むことを特徴とする。

#### 【0009】

本発明によれば、元画像の各画素の奥行き値がインデックス番号として設定されたルックアップテーブルを用いて、仮想オブジェクトに対してインデックスカラー・テクスチャマッピングが行われ、元画像の各画素の奥行き値に応じた値に各画素の $\alpha$ 値が設定される。このように本発明によれば、ゲームシステム（画像生成システム）が元々有しているインデックスカラー・テクスチャマッピングの機能を有効利用して、奥行き値を $\alpha$ 値に変換できる。従って、奥行き値の変換処理を、例えば新たなハードウェアを追加することなく、高速に実行できるようになり、全表示画面分の奥行き値の変換も容易となる。

#### 【0010】

そして本発明によれば、元画像の各画素の奥行き値に応じた値に設定された $\alpha$ 値に基づいて、元画像とぼかし画像とが合成される。従って、奥行き値に応じて、ぼかし画像の合成比率等を変化させることが可能になり、被写界深度などの表現が可能になる。

#### 【0011】

なお、 $\alpha$ （アルファ）値は、各画素に関連づけられて記憶される情報であり、例えば色情報以外の情報である。また奥行き値は、視点（仮想カメラ）から近いほど大きい値にしてもよいし、視点から遠いほど大きい値にしてもよい。また、元画像の合成対象となるぼかし画像の合成手法としては、種々の手法を採用できる。また $\alpha$ 値を用いた合成処理は $\alpha$ ブレンディングに限定されない。

#### 【0012】

また本発明に係るゲームシステム、情報記憶媒体及びプログラムは、元画像の各画素の奥行き値を、前記奥行き値の最上位ビットよりも下位のビットI～ビットJにより構成される第2の奥行き値に変換し、前記第2の奥行き値を、インデックスカラー・テクスチャマッピング用の前記ルックアップテーブルのインデックス番号として設定することを特徴とする。

## 【 0 0 1 3 】

このようにすれば、 $\alpha$  値の実効的な区分け数（奥行き値のしきい値の実効的な段階数）を増やすこと可能となり、例えば、仮想カメラの焦点（注視点）付近にあるオブジェクトのぼけ具合を、多段階のしきい値で精度良く制御できるようになる。従って、生成される画像の品質を高めることができる。

## 【 0 0 1 4 】

また本発明に係るゲームシステム、情報記憶媒体及びプログラムは、前記奥行き値のビット I ～ビット J 以外のビットの値に応じて、前記第 2 の奥行き値を所与の値にクランプすることを特徴とする。

## 【 0 0 1 5 】

このようにすれば、奥行き値のビット I ～ビット J 以外のビットに 1 が立った場合等においても、矛盾の無い画像を生成できる。なお、所与の値としては、第 2 の奥行き値の最大値や最小値、或いは第 2 の奥行き値の上位のビット列を 1 にした値など、種々の値を考えることができる。

## 【 0 0 1 6 】

また本発明に係るゲームシステム、情報記憶媒体及びプログラムは、前記奥行き値を、インデックスカラー・テクスチャマッピング用のルックアップテーブルのインデックス番号として設定し、前記ルックアップテーブルを用いて仮想オブジェクトに対してインデックスカラー・テクスチャマッピングを行い、前記奥行き値を前記第 2 の奥行き値に変換することを特徴とする。

## 【 0 0 1 7 】

このようにすれば、ルックアップテーブルの変換特性を変更するだけという少ない処理負担で、第 2 の奥行き値を所与の値にクランプするなどの多様な変換処理を実現できるようになる。

## 【 0 0 1 8 】

また本発明に係るゲームシステム、情報記憶媒体及びプログラムは、前記奥行き値のビット M ～ビット N を、インデックスカラー・テクスチャマッピング用の第 1 のルックアップテーブルのインデックス番号として設定し、前記第 1 のルックアップテーブルを用いて仮想オブジェクトに対してインデックスカラー・テク

スチャマッピングを行い、前記奥行き値を第3の奥行き値に変換し、前記奥行き値のビットK～ビットL ( $K \geq I \geq L > M \geq J \geq N$ ) を、インデックスカラー・テクスチャマッピング用の第2のルックアップテーブルのインデックス番号として設定し、前記第2のルックアップテーブルを用いて仮想オブジェクトに対してインデックスカラー・テクスチャマッピングを行い、前記奥行き値を第4の奥行き値に変換し、前記第3、第4の奥行き値に基づいて前記第2の奥行き値を求めることを特徴とする。

## 【0019】

このようにすれば、奥行き値の所定範囲のビット列（例えば0～7、8～15、16～23、24～31ビット）しか取り出せないような制限がある場合においても、奥行き値の任意のビットI～Jから構成される第2の奥行き値を得ることができる。これにより、 $\alpha$  値を区分けする奥行き値のしきい値の実効的な段階数を増やすこと可能となり、生成される画像の品質を高めることができる。

## 【0020】

また本発明に係るゲームシステム、情報記憶媒体及びプログラムは、元画像をテクスチャとして設定し、該テクスチャをテクセル補間方式で仮想オブジェクトにマッピングする際に仮想オブジェクトのテクスチャ座標をシフトさせ、元画像のぼかし画像を生成することを特徴とする。

## 【0021】

このようにすれば、テクスチャ座標をシフトさせながらテクセル補間方式で仮想オブジェクトに元画像をマッピングするだけという簡素な処理で、元画像のぼかし画像を生成できるようになる。

## 【0022】

なお、テクセル補間方式とは、特に限定はされないが、テクセルの画像情報を補間してピクセルの画像情報を得る方式などであり、例えば、バイリニアフィルタ方式やトライリニアフィルタ方式などを考えることができる。

## 【0023】

また、仮想オブジェクトは、ポリゴンなどのプリミティブ面であることが望ましいが、立体的なオブジェクトであってもよい。また、仮想オブジェクトは画面

上に表示しないことが望ましいが、表示するようにしてもよい。

【 0 0 2 4 】

また、テクセル補間方式のテクスチャマッピングでほかし画像を生成する場合には、テクスチャ座標を1テクセルよりも小さい値だけシフトさせることが望ましい。また、テクスチャ座標を第1のシフト方向へシフトさせてテクセル補間方式でテクスチャマッピングを行った後に、テクスチャ座標を第2のシフト方向にシフトさせてテクセル補間方式でテクスチャマッピングを行ってもよい。或いは、第1のシフト方向へのシフトと第2のシフト方向へのシフトのセットを、複数回繰り返してもよい。

【 0 0 2 5 】

また本発明に係るゲームシステム、情報記憶媒体及びプログラムは、前記仮想オブジェクトが、表示画面サイズのポリゴンであることを特徴とする。

【 0 0 2 6 】

このようにすれば、全表示画面分の元画像の奥行き値を例えば1回（或いは数回）のテクスチャマッピングで $\alpha$ 値に変換できるようになる。

【 0 0 2 7 】

また本発明に係るゲームシステム、情報記憶媒体及びプログラムは、前記仮想オブジェクトが、表示画面を分割したブロックのサイズのポリゴンであることを特徴とする。

【 0 0 2 8 】

このようにすれば、仮想オブジェクトを描画する領域の大きさを小さくすることが可能になり、記憶部の使用記憶容量を節約できる。

【 0 0 2 9 】

【発明の実施の形態】

以下、本発明の好適な実施形態について図面を用いて説明する。

【 0 0 3 0 】

1. 構成

図1に、本実施形態のゲームシステム（画像生成システム）のブロック図の一例を示す。なお同図において本実施形態は、少なくとも処理部100を含めばよ

く、それ以外のブロックについては、任意の構成要素とすることができる。

【0031】

ここで処理部100は、システム全体の制御、システム内の各ブロックへの命令の指示、ゲーム処理、画像処理、音処理などの各種の処理を行うものであり、その機能は、各種プロセッサ（CPU、DSP等）、或いはASIC（ゲートアレイ等）などのハードウェアや、所与のプログラム（ゲームプログラム）により実現できる。

【0032】

操作部160は、プレーヤが操作データを入力するためのものであり、その機能は、レバー、ボタン、筐体などのハードウェアにより実現できる。

【0033】

記憶部170は、処理部100や通信部196などのワーク領域となるもので、その機能はRAMなどのハードウェアにより実現できる。

【0034】

情報記憶媒体（コンピュータにより使用可能な記憶媒体）180は、プログラムやデータなどの情報を格納するものであり、その機能は、光ディスク（CD、DVD）、光磁気ディスク（MO）、磁気ディスク、ハードディスク、磁気テープ、或いはメモリ（ROM）などのハードウェアにより実現できる。処理部100は、この情報記憶媒体180に格納される情報に基づいて本発明（本実施形態）の種々の処理を行う。即ち情報記憶媒体180には、本発明（本実施形態）の手段（特に処理部100に含まれるブロック）を実行するための情報（プログラム或いはデータ）が格納される。

【0035】

なお、情報記憶媒体180に格納される情報の一部又は全部は、システムへの電源投入時等に記憶部170に転送されることになる。また情報記憶媒体180に記憶される情報は、本発明の処理を行うためのプログラムコード、画像データ、音データ、表示物の形状データ、テーブルデータ、リストデータ、本発明の処理を指示するための情報、その指示に従って処理を行うための情報等の少なくとも1つを含むものである。

## 【 0 0 3 6 】

表示部 1 9 0 は、本実施形態により生成された画像を出力するものであり、その機能は、CRT、LCD、或いはHMD（ヘッドマウントディスプレイ）などのハードウェアにより実現できる。

## 【 0 0 3 7 】

音出力部 1 9 2 は、本実施形態により生成された音を出力するものであり、その機能は、スピーカなどのハードウェアにより実現できる。

## 【 0 0 3 8 】

携帯型情報記憶装置 1 9 4 は、プレーヤの個人データ（セーブデータ）などが記憶されるものであり、この携帯型情報記憶装置 1 9 4 としては、メモリカードや携帯型ゲーム装置などを考えることができる。

## 【 0 0 3 9 】

通信部 1 9 6 は、外部（例えばホスト装置や他のゲームシステム）との間で通信を行うための各種の制御を行うものであり、その機能は、各種プロセッサ、或いは通信用 A S I C などのハードウェアや、プログラムなどにより実現できる。

## 【 0 0 4 0 】

なお本発明（本実施形態）の手段を実行するためのプログラム或いはデータは、ホスト装置（サーバー）が有する情報記憶媒体からネットワーク及び通信部 1 9 6 を介して情報記憶媒体 1 8 0 に配信するようにしてもよい。このようなホスト装置（サーバー）の情報記憶媒体の使用も本発明の範囲内に含まれる。

## 【 0 0 4 1 】

処理部 1 0 0 は、ゲーム処理部 1 1 0、画像生成部 1 3 0、音生成部 1 5 0 を含む。

## 【 0 0 4 2 】

ここでゲーム処理部 1 1 0 は、コイン（代価）の受け付け処理、各種モードの設定処理、ゲームの進行処理、選択画面の設定処理、オブジェクト（1 又は複数のプリミティブ面）の位置や回転角度（X、Y 又は Z 軸回り回転角度）を求める処理、オブジェクトを動作させる処理（モーション処理）、視点の位置（仮想カメラの位置）や視線角度（仮想カメラの回転角度）を求める処理、マップオブジ

ェクトなどのオブジェクトをオブジェクト空間へ配置するための処理、ヒットチェック処理、ゲーム結果（成果、成績）を演算する処理、複数のプレーヤが共通のゲーム空間でプレイするための処理、或いはゲームオーバー処理などの種々のゲーム処理を、操作部160からの操作データや、携帯型情報記憶装置194からの個人データや、ゲームプログラムなどに基づいて行う。

【0043】

画像生成部130は、ゲーム処理部110からの指示等にしたがって各種の画像処理を行い、例えばオブジェクト空間内で仮想カメラ（視点）から見える画像を生成して、表示部190に出力する。また、音生成部150は、ゲーム処理部110からの指示等にしたがって各種の音処理を行い、BGM、効果音、音声などの音を生成し、音出力部192に出力する。

【0044】

なお、ゲーム処理部110、画像生成部130、音生成部150の機能は、その全てをハードウェアにより実現してもよいし、その全てをプログラムにより実現してもよい。或いは、ハードウェアとプログラムの両方により実現してもよい。

【0045】

ゲーム処理部110は、移動・動作演算部112を含む。

【0046】

ここで移動・動作演算部112は、車などのオブジェクトの移動情報（位置データ、回転角度データ）や動作情報（オブジェクトの各パーツの位置データ、回転角度データ）を演算するものであり、例えば、操作部160によりプレーヤが入力した操作データやゲームプログラムなどに基づいて、オブジェクトを移動させたり動作させたりする処理を行う。

【0047】

より具体的には、移動・動作演算部112は、オブジェクトの位置や回転角度を例えば1フレーム（1/60秒）毎に求める処理を行う。例えば（ $k-1$ ）フレームでのオブジェクトの位置を $PM_{k-1}$ 、速度を $VM_{k-1}$ 、加速度を $AM_{k-1}$ 、1フレームの時間を $\Delta t$ とする。すると $k$ フレームでのオブジェクトの位置 $PM$

k、速度VMkは例えば下式(1)、(2)のように求められる。

【0048】

$$PMk = PMk-1 + VMk-1 \times \Delta t \quad (1)$$

$$VMk = VMk-1 + AMk-1 \times \Delta t \quad (2)$$

画像生成部130は、ジオメトリ処理部132、インデックス番号設定部134、描画部140を含む。

【0049】

ここで、ジオメトリ処理部132は、座標変換、クリッピング処理、透視変換、或いは光源計算などの種々のジオメトリ処理(3次元演算)を行う。そして、ジオメトリ処理後(透視変換後)のオブジェクトデータ(オブジェクトの頂点座標などの形状データ、或いは頂点テクスチャ座標、輝度データ等)は、記憶部170の主記憶領域172に保存される。

【0050】

インデックス番号設定部134は、元画像の画像情報(例えば透視変換後の画像の情報)を、LUT(ルックアップテーブル)記憶部178に記憶されるインデックスカラー・テクスチャマッピング用のLUTのインデックス番号として設定するための処理を行う。ここで、画像情報としては、例えば、色情報(RGB、YUV等)、 $\alpha$ 値(各画素に関連づけられて記憶される情報であり色情報以外のプラスアルファの情報)、奥行き値(Z値)等、種々の情報を考えることができる。

【0051】

そして本実施形態では、元画像の各画素の奥行き値がインデックス番号として設定されたルックアップテーブルを用いて、仮想オブジェクトに対してインデックスカラー・テクスチャマッピングを行うことで、元画像の各画素の奥行き値に応じた値に各画素の $\alpha$ 値を設定する。これにより、いわゆる被写界深度の表現が可能になる。

【0052】

なお、仮想カメラの焦点から遠い画素ほど $\alpha$ 値が大きくなるように(広義には、ぼかし画像の合成比率が高くなるように)、LUT(ルックアップテーブル)



における各画素の奥行き値（インデックス番号）と $\alpha$ 値との対応関係を設定することが望ましい。また、LUTにおける各画素の奥行き値と $\alpha$ 値との対応関係を変化させることで、被写界深度の範囲やぼかしエフェクトの強弱を可変に制御してもよい。

#### 【0053】

描画部140は、ジオメトリ処理後のオブジェクト（モデル）を、描画領域174（フレームバッファ、別バッファ等のピクセル単位で画像情報を記憶できる領域）に描画するための処理を行うものであり、テクスチャマッピング部142、 $\alpha$ 合成部144、陰面消去部146を含む。

#### 【0054】

ここでテクスチャマッピング部142は、テクスチャ記憶部176に記憶されるテクスチャをオブジェクトにマッピングするための処理（オブジェクトにマッピングするテクスチャを指定する処理、テクスチャを転送する処理等）を行う。この場合、テクスチャマッピング部142は、LUT記憶部178に記憶されるインデックスカラー・テクスチャマッピング用のLUT（ルックアップテーブル）を用いたテクスチャマッピングを行うことができる。

#### 【0055】

そして本実施形態では、テクスチャマッピング部142が、元画像の画像情報がインデックス番号として設定されたLUTを用いて、仮想オブジェクト（表示画面サイズのポリゴン、分割ブロックサイズのポリゴン等）に対してテクスチャマッピングを行う。これにより、奥行き値（Z値）をNビット化する処理や、奥行き値を $\alpha$ 値に変換する処理や、種々の画像変換処理（ガンマ補正、ネガポジ反転、ボスタリゼーション、ソラリゼーション、2値化、モノトーンフィルタ、セピアフィルタ）を少ない処理負担で実現できるようになる。

#### 【0056】

また本実施形態では、元画像と $\alpha$ 合成（ $\alpha$ ブレンディング、 $\alpha$ 加算、 $\alpha$ 減算、半透明処理等）されるぼかし画像（最もぼけた画像）の生成処理を、テクセル補間方式（バイリニアフィルタ方式、トライリニアフィルタ方式）のテクスチャマッピングを有効利用して実現している。

## 【0057】

即ち、テクスチャマッピング部142は、テクスチャとして設定された元画像を、テクスチャ座標を例えば1ピクセル（テクセル）よりも小さい値だけシフトさせながら（例えば元画像の描画位置に基づき得られるテクスチャ座標からシフトさせながら）、テクセル補間方式で仮想オブジェクト（ぼかし領域と同一形状のオブジェクト）にマッピングする。このようにすれば、テクスチャ座標をシフトさせるだけという簡素な処理で、元画像の合成対象となるぼかし画像を生成できるようになる。

## 【0058】

$\alpha$ 合成部144は、元画像とそのぼかし画像とを、描画領域174（フレームバッファ等）の各画素に対して設定された $\alpha$ 値（A値）に基づいて合成する処理を行う。例えば $\alpha$ 合成が $\alpha$ ブレンディングである場合には、下式のように元画像とぼかし画像とが合成される。

## 【0059】

$$R_Q = (1 - \alpha) \times R_1 + \alpha \times R_2 \quad (3)$$

$$G_Q = (1 - \alpha) \times G_1 + \alpha \times G_2 \quad (4)$$

$$B_Q = (1 - \alpha) \times B_1 + \alpha \times B_2 \quad (5)$$

ここで、 $R_1$ 、 $G_1$ 、 $B_1$ は、描画領域174に既に描画されている元画像の色（輝度）のR、G、B成分であり、 $R_2$ 、 $G_2$ 、 $B_2$ は、ぼかし画像の色のR、G、B成分である。また $R_Q$ 、 $G_Q$ 、 $B_Q$ は、 $\alpha$ ブレンディングにより生成されるR、G、B成分である。

## 【0060】

陰面消去部146は、Z値（奥行き値）が格納されるZバッファ179（Zプレーン）を用いて、Zバッファ法のアルゴリズムにしたがった陰面消去を行う。本実施形態では、このZバッファ179に書き込まれたZ値を $\alpha$ 値に変換し、その $\alpha$ 値に基づいて元画像とぼかし画像の合成処理を行っている。

## 【0061】

なお、本実施形態のゲームシステムは、1人のプレーヤのみがプレイできるシングルプレーヤモード専用のシステムにしてもよいし、このようなシングルプレ

ーヤモードのみならず、複数のプレーヤがプレイできるマルチプレーヤモードも備えるシステムにしてもよい。

---

【 0 0 6 2 】

また複数のプレーヤがプレイする場合に、これらの複数のプレーヤに提供するゲーム画像やゲーム音を、1つの端末を用いて生成してもよいし、ネットワーク（伝送ライン、通信回線）などで接続された複数の端末（ゲーム装置、携帯電話）を用いて生成してもよい。

【 0 0 6 3 】

2. 本実施形態の特徴

2. 1 インデックスカラー・テクスチャマッピングの利用

さて、ゲームシステムにおいて画像を生成する際には、モニタ（表示部）の非線形特性を補正するために、画像に対してガンマ補正と呼ばれる変換を施すことが望ましい。

【 0 0 6 4 】

そして、このようなガンマ補正を実現する手法としては、以下に説明する第1、第2の手法がある。

【 0 0 6 5 】

第1の手法では、図2（A）に示すように、メインメモリ802上にガンマ補正用のLUT（ルックアップテーブル）を用意しておく。そして、CPU800（CPU上で動作するソフトウェア）は、VRAM806内にあるフレームバッファ808から、元画像の各画素の色情報（RGB）を読み出す。そして、読み出された色情報に基づいてガンマ補正用LUTを参照し、ガンマ補正後の色情報を得る。次に、得られたガンマ補正後の色情報をフレームバッファの対応する画素に書き戻す。そして、以上の処理を、元画像の全ての画素に対して行う。

【 0 0 6 6 】

一方、第2の手法では、図2（B）に示すように、CPU810の制御下で動作する描画プロセッサ812の後段に、ハードウェアによりガンマ補正を実現するガンマ補正回路814を設ける。そして、描画プロセッサ812により生成された色情報に対して、ガンマ補正回路814がガンマ補正を施し、モニタ816

に出力する。

【 0 0 6 7 】

しかしながら、図 2 ( A ) の第 1 の手法では、フレームバッファ 8 0 8 からの色情報の読み出し、ガンマ補正用 L U T 8 0 4 の参照、ガンマ補正用 L U T 8 0 4 からの色情報の読み出し、読み出した色情報のフレームバッファ 8 0 8 への書き戻しなどの全ての処理を、C P U 8 0 0 上で動作するソフトウェアが行うことになる。従って、処理の高速化を図れず、表示画面の全画素に対するガンマ補正を、1 フレーム内で完了するのは困難となる。また、C P U 8 0 0 の処理負荷が非常に重くなり、他の処理に悪影響を及ぼすという問題も招く。

【 0 0 6 8 】

一方、図 2 ( B ) の第 2 の手法では、専用のハードウェアであるガンマ補正回路 8 1 4 が使用されるため、高速なガンマ補正が可能になる。従って、表示画面の全画素に対するガンマ補正を 1 フレーム内で完了することも容易となる。また、C P U 8 1 0 の処理負荷も少ないため、他の処理に悪影響が及ぶ問題も解決できる。

【 0 0 6 9 】

しかしながら、図 2 ( B ) の第 2 の手法では、専用のハードウェアであるガンマ補正回路 8 1 4 が別途必要になってしまう。従って、ゲームシステムが大規模化し、製品コストの増加の問題を招く。

【 0 0 7 0 】

特に、家庭用ゲームシステムにおいては、製品の普及化を図るために、低コスト化が厳しく要求されており、ほとんどの家庭用ゲームシステムでは、図 2 ( B ) に示すようなガンマ補正回路がハードウェアとして設けられていない。従って、ガンマ補正を実現するためには、図 2 ( A ) のような第 1 の手法を採用せざるを得ない。

【 0 0 7 1 】

ところが、前述のように第 1 の手法では、1 フレーム内で全表示画面分のガンマ補正を完了するのは困難であり、他の処理にも悪影響を及ぼす。従って、家庭用ゲームシステムにおいては、ガンマ補正の実施自体を断念せざるを得なかった

## 【 0 0 7 2 】

そこで、本出願の発明者は、インデックスカラー・テクスチャマッピングにおいて使用されるルックアップテーブルLUTの存在に着目した。

## 【 0 0 7 3 】

即ち、インデックスカラーテクスチャマッピングでは、テクスチャ記憶部の使用記憶容量を節約するために、図3のA1に示すように、実際の色情報(RGB)ではなくインデックス番号が、テクスチャの各テクセルに関連づけて記憶される。また、図3のA2に示すように、インデックスカラー・テクスチャマッピング用のLUT(カラーパレット)には、インデックス番号により指定される色情報が記憶される。そして、オブジェクトに対してテクスチャマッピングを行う際には、テクスチャの各テクセルのインデックス番号に基づいてLUTを参照し、対応する色情報をLUTから読み出し、読み出された色情報をフレームバッファに描画する。

## 【 0 0 7 4 】

このようなインデックスカラーモードのテクスチャマッピングでは、LUTを用いない通常モードのテクスチャマッピングに比べて、使用できる色数は少なくなる(例えば256色)。しかしながら、テクスチャ記憶部に実際の色情報(例えば16ビットの色情報)を記憶する必要が無くなるため、テクスチャ記憶部の使用記憶容量を大幅に節約できる。

## 【 0 0 7 5 】

本実施形態は、このようなインデックスカラー・テクスチャマッピングを通常とは異なる形態で利用している点に特徴がある。

## 【 0 0 7 6 】

即ち、まず図4のB1に示すように、フレームバッファ(広義には描画領域)に描画されている元画像の各画素の画像情報(例えば色情報)を、ガンマ補正用のルックアップテーブルLUTのインデックス番号として設定する(インデックス番号とみなす)。そしてB2に示すように、元画像の画像情報がインデックス番号として設定されたLUTを用いて、仮想オブジェクト(例えば表示画面サイ

ズのポリゴン) に対してインデックスカラー・テクスチャマッピングを行い、元画像の画像情報を変換する。そしてB3に示すように、変換後の画像情報を、フレームバッファ(描画領域)などに描き戻す。

---

## 【0077】

以上のようにして本実施形態では、図5(A)に示すような元画像から、図5(B)に示すようなガンマ補正が施された画像を得ることに成功している。即ち図5(B)の画像では、図5(A)に比べて、よりコントラストのはっきりした画像になっている。

## 【0078】

例えば図2(A)の第1の手法では、元画像の色情報の読み出し、ガンマ補正用LUTの参照、色情報のフレームバッファへの書き戻しなどの全ての処理をCPU上で動作するソフトウェアが行うことになるため、処理の高速化を図れないと共に、CPUの処理負荷も過大になる。

## 【0079】

これに対して本実施形態では、インデックスカラー・テクスチャマッピングを有効利用してガンマ補正を実現しており、このインデックスカラー・テクスチャマッピングは、専用のハードウェアである描画プロセッサ(描画部)により高速に実行される。従って本実施形態によれば、図2(A)の第1の手法に比べて高速にガンマ補正を実行でき、全表示画面分のガンマ補正を1フレーム(例えば1/60秒、1/30秒)内で完了することも容易となる。

---

## 【0080】

また、インデックスカラー・テクスチャマッピングは、メインプロセッサ(CPU)とは独立に動作する描画プロセッサにより実行できるため、メインプロセッサ(CPU)の処理負荷の増加も最小限に抑えることができる。従って、ガンマ補正の実行が要因となって他の処理に悪影響が及ぶ事態も防止できる。

## 【0081】

また、従来のゲームシステムでは、描画プロセッサの処理能力はそれほど高くなかった。従って、フレームバッファへの元画像の描画と、表示画面サイズのポリゴンの描画を、1フレーム内で完了させることは難しかった。

## 【0082】

しかしながら、ゲームシステムにおいて、描画プロセッサの処理能力の向上は、他の回路ブロックの処理能力の向上に比べて著しく大きく、非常に高いフレームレート（1秒間にレンダリングできるテクセル数）を持つ描画プロセッサがゲームシステムに使用されるようになってきた。従って、フレームバッファへの元画像の描画と、表示画面サイズのポリゴンの描画を、1フレーム内で完了させることも容易となり、インデックスカラー・テクスチャマッピングを有効利用したガンマ補正も無理なく実現できるようになってきた。

## 【0083】

また図2（B）の第2の手法では、専用のハードウェアであるガンマ補正回路が別途必要になり、ゲームシステムの高コスト化を招く。また、このようなガンマ補正回路を元々有しない家庭用ゲームシステムなどでは、図2（B）に示す第2の手法を実現することはできず、図2（A）の手法を採用せざるを得なかった。

## 【0084】

これに対して本実施形態では、インデックスカラー・テクスチャマッピングを有効利用してガンマ補正を実現しており、このインデックスカラー・テクスチャマッピングは、描画プロセッサが元々持っているハードウェアにより実行される。従って本実施形態によれば、図2（B）のようなガンマ補正回路を新たに付加する必要がなく、ゲームシステムが高コスト化してしまう事態を防止できる。また、ガンマ補正回路を元々有していない家庭用ゲームシステムにおいても、ハードウェアによる高速なガンマ補正を実現できるようになる。

## 【0085】

なお、図4では表示画面サイズのポリゴンにテクスチャマッピングしてガンマ補正（ビデオフィルタ）を実現しているが、表示画面を分割したブロックのサイズのポリゴンにテクスチャマッピングするようにしてもよい。

## 【0086】

即ち、図6のC1に示すように、フレームバッファ上の元画像（表示画面）を複数のブロックに分割し、C2に示すように、各ブロックの画像を、LUTを用

いて分割ブロックサイズのポリゴンにテクスチャマッピングする。そして、得られた分割ブロックサイズの画像をフレームバッファ（描画領域）に描き戻す。

---

【0087】

或いは、透視変換後（スクリーン座標系への変換後）のオブジェクトの画像の全部又は一部を内包し、透視変換後のオブジェクトの大きさに応じてその大きさが変化するようなポリゴン（仮想オブジェクト）を生成し、そのポリゴンにテクスチャマッピングする。

【0088】

このようにすれば、例えばテクスチャマッピングされたポリゴンを別バッファに一時的に描画するような場合に、VRAM上での別バッファの占有領域を小さくできる。

【0089】

即ち、図4のように表示画面サイズのポリゴンにテクスチャマッピングすると、この表示画面サイズのポリゴンを一時的に描画するために、表示画面サイズの別バッファをVRAM上に確保しなければならず、他の処理に支障を来すおそれがある。

【0090】

図6のように、分割ブロックサイズのポリゴンにテクスチャマッピングするようになれば、VRAM上には分割ブロックサイズの別バッファを用意すれば済むため、別バッファの占有領域を小さくできる。従って、限られたハードウェア資源を有効利用することが可能になる。

【0091】

## 2. 2 マスク処理

さて、ガンマ補正では、1つの入力値（RIN、GIN又はBIN）に対して1つの値（ROUT、GOUT又はBOUT）が出力されるようなLUTが必要になる。

【0092】

ところが、図3に示すインデックスカラー・テクスチャマッピング用のLUTは、元々、ガンマ補正用に設計されたものではないため、1つ入力値（インデッ



クス番号) に対して、複数の値 (例えば R O U T、G O U T、及び B O U T) が出力されてしまう。従って、この L U T の不整合を解決しなければならないという課題がある。

#### 【 0 0 9 3 】

そこで本実施形態では、元画像の画像情報 (R、G、B、Z 値又は  $\alpha$  値等) を L U T のインデックス番号として設定する場合において、変換により得られた画像情報のうち必要な画像情報のみが描画領域 (フレームバッファ、別バッファ) に描画され、他の画像情報が描画されないようにするためのマスク処理を行っている。

#### 【 0 0 9 4 】

より具体的には、図 7 の D 1 に示すように、元画像の R プレーンの値をインデックス番号に設定して、L U T を用いたテクスチャマッピングを行うと、R (R O U T)、G (G O U T)、B (B O U T) という 3 つプレーンの値が出力される。そして、この場合には、D 2 に示すように、出力された R プレーンの値のみを描画領域に描画し、G プレーン、B プレーンの値についてはマスクして描画領域に描画されないようにする。

#### 【 0 0 9 5 】

また、図 7 の D 3 に示すように、元画像の G プレーンの値をインデックス番号に設定してテクスチャマッピングを行った場合には、D 4 に示すように、出力された G プレーンの値のみを描画領域に描画し、R プレーン、B プレーンの値についてはマスクして描画領域に描画されないようにする。

#### 【 0 0 9 6 】

また、図 7 の D 5 に示すように、元画像の B プレーンの値をインデックス番号に設定してテクスチャマッピングを行った場合には、出力された B プレーンの値のみを描画領域に描画し、R プレーン、G プレーンの値についてはマスクして描画領域に描画されないようにする。

#### 【 0 0 9 7 】

以上のようにすることで、元々はガンマ補正用に設計されていないインデックスカラー・テクスチャマッピング用の L U T を用いながらも、少ない処理負荷で

、元画像の画像変換処理を実行できるようになる。

【0098】

---

2. 3 Z 値、 $\alpha$  値への利用

以上では、インデックスカラー・テクスチャマッピング用の LUT に基づき出力される色情報 R、G、B を利用する場合について説明した。

【0099】

しかしながら、インデックスカラー・テクスチャマッピング用の LUT に基づき出力される  $\alpha$  値（A 値。画素に関連づけて設定される色情報以外の情報）を利用するようにしてもよい。

【0100】

例えば図 8 に示すように、R プレーン（又は G、B）の値をインデックス番号として設定して、LUT を用いたテクスチャマッピングを行い、 $\alpha$ （ $\alpha$  OUT）プレーンを生成する。そして、生成された  $\alpha$  プレーンを用いて、マスク処理などを行うようにする。

【0101】

即ち、例えば R 値が 0～127 の時には  $\alpha$  値（ $\alpha$  OUT）が 0 になり、R 値が 128～255 の時には  $\alpha$  値が 255 になるように  $\alpha$  値が設定された LUT を使用する。そして、 $\alpha$  値が 255 よりも小さい画素についてはマスク処理を行わず、 $\alpha$  値が 255 の画素についてはマスク処理を行うようにする。このようにすれば、R 値が 128 以上の画素に対してだけマスク処理が行われるようになり、各画素の R 値の大小に応じたマスク処理を行うことができるようになる。

【0102】

なお、生成された  $\alpha$  プレーンの値を  $\alpha$  合成の係数（透明度、半透明度、不透明度）として使用してもよい。

【0103】

また、LUT のインデックス番号として設定される画像情報は色情報に限定されない。即ち、描画領域（VRAM）上にあり、LUT のインデックス番号として設定できる画像情報であればよい。

【0104】

例えば図9に示すように、Z値（奥行き値）をLUTのインデックス番号として設定するようにしてもよい。

---

【0105】

そして、この場合には、Z値をインデックス番号に設定してインデックスカラー・テクスチャマッピングを行うことで得られる $\alpha$ プレーンの値を、例えば $\alpha$ 合成の係数として使用するようにする。このようにすることで、Z値に応じた値の $\alpha$ 値を設定できるようになり、ぼかし画像を用いた被写界深度などの表現が可能になる。

【0106】

即ち図9に示すようなLUTを用いたテクスチャマッピングを行うことで、図10のF1に示すように、元画像の各画素A、B、C、DのZ値 $Z_A$ 、 $Z_B$ 、 $Z_C$ 、 $Z_D$ に応じた値に各画素の $\alpha$ 値 $\alpha_A$ 、 $\alpha_B$ 、 $\alpha_C$ 、 $\alpha_D$ を設定する。そして、例えば図10のF2に示すような $\alpha$ プレーンを生成する。より具体的には、仮想カメラ10の焦点（注視点）から遠い画素（焦点とのZ値の差が大きい画素）ほど、例えば大きな $\alpha$ 値を設定する。これにより、仮想カメラ10の焦点から遠い画素ほど、ぼかし画像の合成比率が高くなる。

【0107】

そして、図10のF3に示すように、生成された $\alpha$ プレーン（各画素に設定された $\alpha$ 値）に基づいて、元画像とぼかし画像の $\alpha$ 合成（ $\alpha$ ブレンディング等）を行う。なお、図11（A）に元画像の例を示し、図11（B）に、そのぼかし画像の例を示す。

【0108】

このように、Z値（奥行き値）に応じて設定された $\alpha$ 値に基づき元画像（図11（A））とぼかし画像（図11（B））の $\alpha$ 合成を行うことで、例えば、仮想カメラの焦点（ピントが合っている点として設定される点）から遠くなるほどぼけて見える画像を生成できるようになり、いわゆる被写界深度の表現が可能になる。これにより、画面内の全ての被写体にピントが合っていた従来のゲーム画像とは異なり、現実世界の視界画像のように視点からの距離に応じてフォーカシングされたリアルで自然なゲーム画像を生成できる。この結果、プレーヤの仮想現

実感を格段に向上できる。

【0 1 0 9】

例えば図 1 2 に、Z 値に応じた  $\alpha$  値の設定の一例を示す。なお、図 1 2 では、 $\alpha$  値は、その大きさが 1. 0 以下になるように正規化されている。

【0 1 1 0】

そして図 1 2 では、Z 値  $Z_1 \sim Z_4$ 、 $Z_1' \sim Z_4'$  (しきい値) により領域  $AR_0 \sim AR_4$ 、 $AR_1' \sim AR_4'$  の区分けが行われる。そして、これらの領域  $AR_0 \sim AR_4$ 、 $AR_1' \sim AR_4'$  に対して、 $\alpha$  値  $\alpha_0 \sim \alpha_4$ 、 $\alpha_1' \sim \alpha_4'$  が設定される。

【0 1 1 1】

例えば、 $Z_1 \sim Z_2$  の間の領域  $AR_1$  にある画素については、その  $\alpha$  値が  $\alpha_1$  に設定され、 $Z_2 \sim Z_3$  の間の領域  $AR_2$  にある画素については、その  $\alpha$  値が  $\alpha_2$  に設定される。また、 $Z_1' \sim Z_2'$  の間の領域  $AR_1'$  にある画素については、その  $\alpha$  値が  $\alpha_1'$  に設定され、 $Z_2' \sim Z_3'$  の間の領域  $AR_2'$  にある画素については、その  $\alpha$  値が  $\alpha_2'$  に設定される。

【0 1 1 2】

そして、各領域に設定される  $\alpha$  値には例えば以下の関係式が成り立つ。

【0 1 1 3】

$$\alpha_0 < \alpha_1 < \alpha_2 < \alpha_3 < \alpha_4 \quad (6)$$

$$\alpha_0 < \alpha_1' < \alpha_2' < \alpha_3' < \alpha_4' \quad (7)$$

これらの式 (6)、(7) から明らかなように、仮想カメラ 1 0 の焦点 (注視点) から遠いほど  $\alpha$  値が大きくなっている。即ち、仮想カメラ 1 0 の焦点との Z 値の差が大きい画素ほど、ぼかし画像の合成比率が高くなるように、 $\alpha$  値が設定されている。

【0 1 1 4】

このように  $\alpha$  値を設定することで、仮想カメラの焦点から遠くなるほどぼけて見える画像を生成でき、いわゆる被写界深度の表現が可能になる。

【0 1 1 5】

しかも本実施形態では、LUT を用いた 1 回のテクスチャマッピングで、各画

素のZ値を $\alpha$ 値に変換できるため、処理負担が非常に軽いという利点がある。

【0116】

即ち、LUTを用いない $\alpha$ 値の設定手法の例として、図13(A)、(B)、(C)に示す手法が考えられる。

【0117】

この手法では図13(A)に示すように、まず、Z値がZ<sub>1</sub>に設定される仮想オブジェクトOB<sub>1</sub>(ポリゴン)を、フレームバッファに描画することで、仮想オブジェクトOB<sub>1</sub>を基準に奥側にある画素の $\alpha$ 値を更新する。即ち、Z値に基づく陰面消去手法を有効利用して、OB<sub>1</sub>よりも奥側にある画素の $\alpha$ 値を更新する。

【0118】

次に、図13(B)に示すように、Z値がZ<sub>2</sub>に設定される仮想オブジェクトOB<sub>2</sub>をフレームバッファに描画することで、仮想オブジェクトOB<sub>2</sub>を基準に奥側にある画素の $\alpha$ 値を更新する。同様に、図13(C)に示すように、Z値がZ<sub>3</sub>に設定される仮想オブジェクトOB<sub>3</sub>をフレームバッファに描画することで、仮想オブジェクトOB<sub>3</sub>を基準に奥側にある画素の $\alpha$ 値を更新する。

【0119】

このような手法を採用すれば、領域AR<sub>1</sub>にある画素の $\alpha$ 値を $\alpha_1$ に設定し、領域AR<sub>2</sub>にある画素の $\alpha$ 値を $\alpha_2$ に設定し、領域AR<sub>3</sub>にある画素の $\alpha$ 値を $\alpha_3$ に設定できるようになる。即ち、LUTを用いなくても、各画素のZ値に応じた値に各画素の $\alpha$ 値を設定できる。

【0120】

しかしながら、この手法では、Z値のしきい値の段階数分だけ、仮想オブジェクトの描画処理が必要になる。例えば図12の場合は8回の描画処理が必要になる。従って、この手法には描画処理の負担が重くなるという欠点がある。一方、描画処理の負担を軽減するためにZ値のしきい値の段階数を減らすと、今度は、Z値のしきい値の境界が表示画面上で帯状に見えてしまう事態が生じ、表示品質が低下する。

【0121】

LUTを用いてZ値を $\alpha$ 値に変換する本実施形態の手法によれば、LUTを用いた1回のテクスチャマッピングで、各画素のZ値を $\alpha$ 値に一斉に変換できる。

例えば、LUTのインデックス番号（エントリー）のビット数が8ビットの場合には、LUTを用いた1回のテクスチャマッピングで、256段階のZ値のしきい値で分けられる $\alpha$ 値を得ることができる。従って、Z値のしきい値の境界が表示画面上で帯状に見えてしまう事態を防止でき、少ない処理負担で高品質な画像を生成できる。

【0122】

## 2. 4 Z値の8ビット化

さて、ゲームシステムにおいては、陰面消去の精度を高めるために、Z値のビット数は例えば24ビット、32ビットというように非常に多い。

【0123】

一方、インデックスカラー・テクスチャマッピング用のLUTのインデックス番号のビット数（LUTのエントリーのビット数）は、例えば8ビットというようにZ値のビット数に比べて少ない。

【0124】

従って、図9のようにインデックスカラー・テクスチャマッピング用のLUTを用いてZ値を $\alpha$ 値に変換する場合には、その前処理として、Z値を、LUTのインデックス番号のビット数と同じビット数のZ値（以下、Z2値と呼ぶ）に変換する処理が必要になる。即ち、LUTのインデックス番号のビット数が8ビットである場合には、Z値を8ビットのZ2値（第2の奥行き値）に変換する処理が必要になる。

【0125】

この場合に、矛盾の無いZ2値を得るためには、Z値の最上位ビットを含む上位の8ビットをビットセレクトしたものをZ2値とする必要がある。即ち図14において、ビット23（最上位ビット）～16の8ビットをビットセレクトして、Z2値として設定する。

【0126】

しかしながら、このようにZ値の上位ビット23～16の8ビットをZ2値に

設定し、このZ 2 値をLUTを用いて $\alpha$  値に変換すると、 $\alpha$  値の区分け数が少なくなってしまうということが判明した。

#### 【0127】

例えば図15のように、Z 値が4ビットであり、Z 値の上位の2ビットをセレクトしてZ 2 値に設定し、このZ 2 値を $\alpha$  値に変換した場合を考える。この場合には、Z = 0 ~ 15 の全範囲において4 段階のしきい値で $\alpha$  値が区分けされることになる。

#### 【0128】

しかしながら、図15のOB1のようにスクリーンSC（透視変換面）の手前にあるオブジェクトについては、通常はニアクリップされてしまう。そして、例えば図15においてZ = 10 ~ 15 の範囲にあるオブジェクトがニアクリップされるような場合には、Z 値の最上位ビットに1 が立つことは希となる。また、OB2のようにスクリーンSCの近くにあるオブジェクトについては、何れにせよ最もぼけた画像になるため、ぼけ具合を精度良く制御する必要がない。従って、図15のG1に示す部分における2 段階の区分けは無駄になる。

#### 【0129】

そこで本実施形態では図14に示すように、元画像の各画素のZ 値（奥行き値）を、Z 値の最上位ビットよりも下位のビットI ~ J（例えばビット19 ~ 12）により構成されるZ 2 値（第2の奥行き値）に変換し、このZ 2 値を、インデックスカラー・テクスチャマッピング用のLUTのインデックス番号として設定してテクスチャマッピングを行い、各画素の $\alpha$  値を求める。そして、この求められた各画素の $\alpha$  値に基づいて、元画像とぼかし画像を $\alpha$  合成する。

#### 【0130】

このようにすれば図16に示すように、仮想カメラ10の焦点（注視点）付近にあるオブジェクト（例えばOB3、OB4）についてだけ、多段階（図16では4 段階）のしきい値で区分けされた $\alpha$  値により、ぼけ具合を精度良く制御できるようになる。従って、生成される画像の品質を高めることができる。

#### 【0131】

そして本実施形態では、Z 値のビットI ~ J以外のビットの値に応じて、Z 2

値を所与の値にクランプするようにしている。より具体的には図16に示すように、Z値の上位のビットに例えば1が立った場合には、Z2値を最大値（広義には所与の値）にクランプする。このようにすれば、OB1やOB2のように、ぼけ具合を精度良く制御する必要がないオブジェクトについては、Z2値が最大値に設定され、最もぼけた画像に設定される。従って、Z値を、そのビットI～Jにより構成されるZ2値に変換しても、矛盾の無い画像を生成できる。

【0132】

## 2. 5 LUTを利用したZ値の変換処理

さて本実施形態では、図14で説明したZ値をZ2値に変換する処理（8ビット化処理）を、インデックスカラー・テクスチャマッピング用のLUTを利用したテクスチャマッピングにより実現している。即ち、Z値をLUTのインデックス番号として設定し、そのLUTを用いて仮想オブジェクトに対してインデックスカラー・テクスチャマッピングを行い、Z値をZ2値に変換する。

【0133】

例えば図17に示すように24ビットのZ値をLUTを用いて変換する場合を考える。この場合には図17のH1に示すように、Z値のビット15～8（ビットM～N）をLUT1（第1のルックアップテーブル）のインデックス番号に設定し、LUT1を用いてインデックスカラー・テクスチャマッピングを行い、Z値をZ3値（第3の奥行き値）に変換する。

【0134】

次に図17のH2に示すように、Z値のビット23～16（ビットK～L）をLUT2（第2のルックアップテーブル）のインデックス番号に設定し、LUT2を用いてインデックスカラー・テクスチャマッピングを行い、Z値をZ4値（第4の奥行き値）に変換する。

【0135】

そして図17のH4に示すようにZ3値とZ4値とに基づいてZ2値を求め、H5に示すようにこのZ2値をLUT3（第3のルックアップテーブル）を用いて $\alpha$ 値に変換する。

【0136】



より具体的には、LUT1の変換により得られたZ3値を描画領域（フレームバッファ、別バッファ）に描画する。その後、LUT2の変換により得られたZ4値を描画領域に描画する。この際、Z3値の下位の4ビット（有効ビット）をマスクして、これらのビットにZ4値が上書きされないようにして、Z2値を求める。

## 【0137】

図17の手法を採用することで、Z値の任意の8ビット（広義には任意のビットI～J）を取り出すことが可能になる。

## 【0138】

即ち、LUTのインデックス番号に設定するためにZ値の8ビットを取り出そうとした場合に、Z値のビット23～16、15～8又は7～0というように所定範囲の8ビットしか取り出せない場合がある。

## 【0139】

一方、Z値の中のどの8ビットを取り出すかは、図14、図15で説明したように、ニアクリップの範囲や、仮想カメラの焦点（注視点）の位置などに応じて決められる。

## 【0140】

従って、Z値のビット23～16、15～8又は7～0というように所定範囲の8ビットしか取り出せないと、仮想カメラの焦点付近で最も精度良くぼけ具合を制御できるような適切な $\alpha$ 値を得ることができない。

## 【0141】

例えば、Z値のビット19～12をZ2値として取り出せば、 $\alpha$ 値の実効的な区分け数（Z値のしきい値数）を256段階にできる場合を考える。このような場合に、Z値のビット23～16や15～8しか取り出せないと、 $\alpha$ 値の実効的な区分け数が例えば16段階になってしまい、画質が低下する。

## 【0142】

これに対して、図17に示す手法を採用すれば、上記のようにZ値の所定範囲の8ビットしか取り出せないような場合にも、Z値の任意のビットI～JをZ2値として取り出すことができるようになる。従って、ニアクリップの範囲や、仮

想カメラの焦点の位置などに応じて、 $\alpha$  値を区分けする Z 値のしきい値の設定を最適なものにすることが可能になり、より高品質な画像を生成できるようになる

#### 【0143】

なお図18、図19、図20(A)に、LUT1、LUT2、LUT3の具体例を示す。また図20(B)に、Z2値を $\alpha$ 値に変換するLUT3の変換特性曲線の例を示す。

#### 【0144】

図18に示すようにLUT1は、インデックス番号として入力されたZ値のビット15～8(ビットM～N)を4ビットだけ右にシフトする変換を行う。例えば0x10(16進数表現)は0x01に変換され、0x20は0x02に変換される。

#### 【0145】

また図19に示すようにLUT2は、インデックス番号として入力されたZ値のビット23～16(ビットK～L)を4ビットだけ左にシフトする変換を行う。例えば0x01は0x10に変換され、0x02は0x20に変換される。

#### 【0146】

そして図19のQ1に示すように、入力されたZ値が0x0Fよりも大きい場合には、LUT2の出力は0xF0にクランプされる。

#### 【0147】

Z値がクランプされた場合の例を図21に示す。図21に示すように、ビット20(ビットI～J以外のビット)に1が立った場合には、LUT2の出力が0xF0にクランプされる。これによりZ2値は0xF1になる。

#### 【0148】

例えばLUT2の出力をクランプせずに、Z値のビット19～12をそのまま取り出してしまうと、ビット20に1が立っているにもかかわらず、Z2値は0x11になってしまい、被写界深度の設定が誤った設定になってしまう問題が生じる。

#### 【0149】

LUT2の出力をクランプするようにすれば、このような問題が生じるのを防止でき、被写界深度の設定を適正なものにすることができる。

---

【0150】

しかも、このようにLUT2の出力をクランプしても、図16から明らかなように、ニアクリップされるオブジェクトOB1やスクリーンSCの近くのオブジェクトOB2のぼけ度合いが最大値に設定されるだけであるため、画像が不自然になることもない。

【0151】

2. 6 ぼかし画像の生成

さて、本実施形態では、テクスチャマッピングのバイリニアフィルタ方式（テクセル補間方式）を有効利用して、元画像（図11（A））の合成対象となるぼかし画像（図11（B））を生成している。

【0152】

即ち、テクスチャマッピングにおいては画素の位置とテクセルの位置がずれる場合がある。

【0153】

この場合に、図22に示すように、ポイントサンプリング方式では、画素（サンプリング点）Pの色CP（広義には画像情報）は、Pに最も距離が近いテクセルTAの色CAになる。

【0154】

一方、バイリニアフィルタ方式では、Pの色CPは、Pの周りのテクセルTA、TB、TC、TDの色CA、CB、CC、CDを補間した色になる。

【0155】

より具体的には、TA～TDの座標とPの座標とに基づき、X軸方向の座標比 $\beta : 1 - \beta$  ( $0 \leq \beta \leq 1$ ) と、Y軸方向の座標比 $\gamma : 1 - \gamma$  ( $0 \leq \gamma \leq 1$ ) を求める。

【0156】

この場合に、Pの色CP（バイリニアフィルタ方式での出力色）は、下式のようになる。

【0157】

$$CP = (1 - \beta) \times (1 - \gamma) \times CA + \beta \times (1 - \gamma) \times CB \\ + (1 - \beta) \times \gamma \times CC + \beta \times \gamma \times CD \quad (8)$$

本実施形態では、このようにバイリニアフィルタ方式では色が自動的に補間されることに着目して、ぼかし画像を生成している。

【0158】

より具体的には図23のR1に示すように、例えばフレームバッファに描画されている元画像をテクスチャとして設定する。そして、このテクスチャ（元画像）を仮想オブジェクトにバイリニアフィルタ方式でマッピングする際に、仮想オブジェクトの頂点に与えるテクスチャ座標を、例えば（0.5、0.5）だけ右下方向にシフト（ずらす、移動）させる。このようにすることで、バイリニアフィルタ方式の補間機能により自動的に、元画像の画素の色が周囲ににじんだようなぼかし画像を生成できるようになる。

【0159】

なお、画面全体をぼかす場合には、テクスチャ（元画像）をマッピングする仮想オブジェクトの形状は、画面（ぼかし領域）と同一形状に設定される。即ち、画面の頂点座標が（X、Y）＝（0、0）、（640、0）、（640、480）、（0、480）であった場合には、仮想オブジェクトの頂点座標も（X、Y）＝（0、0）、（640、0）、（640、480）、（0、480）になる。

【0160】

そして、この場合に、仮想オブジェクトの頂点VX1、VX2、VX3、VX4に与えるテクスチャ座標（U、V）を、各々、（0、0）、（640、0）、（640、480）、（0、480）に設定すれば、画面の画素の位置とテクスチャのテクセルの位置とがずれずに一致する。従って、画像はぼけない。

【0161】

これに対して、仮想オブジェクトの頂点VX1、VX2、VX3、VX4に与えるテクスチャ座標（U、V）を、各々、（0.5、0.5）、（640.5、0.5）、（640.5、480.5）、（0.5、480.5）に設定すれば、画面の画素の位置とテクスチャのテクセルの位置とがずれるようになる。従って

、バイリニアフィルタ方式の補間機能により、色の補間が行われ、画像がぼけて見えるようになる。

#### 【0162】

なお、画面の一部の領域をぼかす場合には、仮想オブジェクトの形状を、そのぼかし領域と同一形状にすればよい。

#### 【0163】

また本実施形態では、図24のR3に示すように、元画像をテクスチャに設定し、例えば右下方向（第1のシフト方向）に0.5テクセルだけシフトしてバイリニアフィルタ方式でテクスチャマッピングを行い、第1のぼかし画像を生成する。次に、図24のR4に示すように、この第1のぼかし画像をテクスチャに設定し、例えば左上方向（第2のシフト方向）に0.5テクセルだけシフトしてバイリニアフィルタ方式でテクスチャマッピングを行い、第2のぼかし画像を生成する。或いは、以上の処理（右下方向のシフトと左上方向のシフト）を複数回繰り返す。このようにすることで、更に自然でぼかし効果の強いぼかし画像を生成できるようになる。

#### 【0164】

次に、バイリニアフィルタ方式の補間機能によりぼかし画像が生成される原理について説明する。

#### 【0165】

例えば図25（A）に示すように、テクスチャ座標を0.5テクセルだけ右下方向にシフトさせて、バイリニアフィルタ方式のテクスチャマッピングを行ったとする。この場合には、上式（8）において $\beta = \gamma = 1/2$ になるため、テクセルT44、T45、T54、T55の色をC44、C45、C54、C55とすると、画素P44の色CP44は下式のようなになる。

#### 【0166】

$$CP44 = (C44 + C45 + C54 + C55) / 4 \quad (9)$$

以上から明らかなように、図25（A）に示す変換により、テクセルT44の色C44（変換前の元画像の画素P44の元の色に相当）は、周りの画素P33、P34、P43、P44に対して1/4ずつしみ出すことになる。

## 【0167】

そして、その後に図25(B)に示すように、図25(A)で得られた画像をテクスチャとして、テクスチャ座標を0.5テクセルだけ左上方向にシフトさせてバイリニアフィルタ方式でテクスチャマッピングを行ったとする。この場合には、図25(A)の画素P33、P34、P43、P44が、図25(B)のテクセルT33、T34、T43、T44に対応するようになる。そして、図25(A)でP33、P34、P43、P44(T33、T34、T43、T44)に対して $1/4$ ずつしみ出した色C44が、更に $1/4$ 倍されて周りの4つの画素に対してしみ出すことになる。即ち、結局、元のT44の色C44が $1/4 \times 1/4 = 1/16$ ずつ周りにしみ出すことになる。

## 【0168】

従って、図25(A)、(B)の変換により、画素P33、P34、P35には、各々、色C44(フレームバッファに描かれた元画像の画素P44の元の色に相当)が $1/16$ 、 $2/16$ 、 $1/16$ ずつしみ出すことになる。また、画素P43、P44、P45には、各々、色C44が $2/16$ 、 $4/16$ 、 $2/16$ ずつしみ出し、画素P53、P54、P55には、各々、色C44が $1/16$ 、 $2/16$ 、 $1/16$ ずつしみ出すことになる。

## 【0169】

従って、図25(A)、(B)の変換により、結局、図26(A)に示すような平面フィルタが元画像に対して施されるようになる。この平面フィルタによれば、元画像の各画素の色がその周りに均一に広がるようになり、元画像の理想的なぼかし画像を生成できる。

## 【0170】

また、図25(A)、(B)の変換のセットを2回行えば、図26(B)に示すような平面フィルタが元画像に対して施されるようになる。この平面フィルタによれば、図26(A)よりも更に理想的なぼかし画像を生成できる。

## 【0171】

## 3. 本実施形態の処理

次に、本実施形態の処理の詳細例について、図27、図28のフローチャート

を用いて説明する。

【0172】

まず、元画像（透視変換後の画像）をフレームバッファに描画する（ステップ S 1）。この際に、Z バッファには各画素の Z 値が書き込まれることになる。

【0173】

次に、Z バッファの Z 値のビット 15～8 を変換する LUT 1（図 18）、ビット 23～16 を変換する LUT 2（図 19）、8 ビット化された Z 値を  $\alpha$  値（A 値）に変換する LUT 3（図 20（A））を、VRAM に転送する（ステップ S 2）。

【0174】

次に、Z 値のビット 15～8 を LUT 1 のインデックス番号に設定し、LUT 1 を用いて仮想ポリゴンにテクスチャマッピングを行い、その仮想ポリゴンを別バッファに描画する（ステップ S 3）。

【0175】

次に、Z 値のビット 23～16 を LUT 2 のインデックス番号に設定し、LUT 2 を用いて仮想ポリゴンにテクスチャマッピングを行い、その仮想ポリゴンを別バッファに描画する（ステップ S 4）。この際、図 17 で説明したように、8 ビット化された Z 値の下位の 4 ビット（データ有効ビット）については上書きされないようにマスクしておく。

【0176】

次に、ステップ S 4 で得られた 8 ビットの Z 2 値を LUT 3 のインデックス番号に設定し、LUT 3 を用いて仮想ポリゴンにテクスチャマッピングを行い、その仮想ポリゴンをフレームバッファ（ $\alpha$  プレーン）に描画する（ステップ S 5）。

【0177】

次に、ステップ S 1 でワークバッファに描画された元画像を、テクスチャ座標 U、V を（0.5、0.5）だけシフトしてバイリニアフィルタ方式で仮想ポリゴンにマッピングしながら、その仮想ポリゴンを別バッファに描画する（ステップ S 6）。

【0178】

次に、ステップS6で別バッファに描画された画像を、テクスチャ座標U、Vを $(-0.5, -0.5)$ だけシフトしてバイリニアフィルタ方式で仮想ポリゴンにマッピングしながら、その仮想ポリゴンをフレームバッファに描画する（ステップS7）。この際、ステップS5でフレームバッファに描画された $\alpha$ 値を用いて $\alpha$ ブレンディングを行い、元画像とぼかし画像の $\alpha$ 合成を行う。

【0179】

以上のようにして、いわゆる被写界深度の表現が可能になる。

【0180】

#### 4. ハードウェア構成

次に、本実施形態を実現できるハードウェアの構成の一例について図29を用いて説明する。

【0181】

メインプロセッサ900は、CD982（情報記憶媒体）に格納されたプログラム、通信インターフェース990を介して転送されたプログラム、或いはROM950（情報記憶媒体の1つ）に格納されたプログラムなどに基づき動作し、ゲーム処理、画像処理、音処理などの種々の処理を実行する。

【0182】

コプロセッサ902は、メインプロセッサ900の処理を補助するものであり、高速並列演算が可能な積和算器や除算器を有し、マトリクス演算（ベクトル演算）を高速に実行する。例えば、オブジェクトを移動させたり動作（モーション）させるための物理シミュレーションに、マトリクス演算などの処理が必要な場合には、メインプロセッサ900上で動作するプログラムが、その処理をコプロセッサ902に指示（依頼）する。

【0183】

ジオメトリプロセッサ904は、座標変換、透視変換、光源計算、曲面生成などのジオメトリ処理を行うものであり、高速並列演算が可能な積和算器や除算器を有し、マトリクス演算（ベクトル演算）を高速に実行する。例えば、座標変換、透視変換、光源計算などの処理を行う場合には、メインプロセッサ900で動



作するプログラムが、その処理をジオメトリプロセッサ904に指示する。

【0184】

データ伸張プロセッサ906は、圧縮された画像データや音データを伸張するデコード処理を行ったり、メインプロセッサ900のデコード処理をアクセレートする処理を行う。これにより、オープニング画面、インターミッション画面、エンディング画面、或いはゲーム画面などにおいて、MPEG方式等で圧縮された動画像を表示できるようになる。なお、デコード処理の対象となる画像データや音データは、ROM950、CD982に格納されたり、或いは通信インターフェース990を介して外部から転送される。

【0185】

描画プロセッサ910は、ポリゴンや曲面などのプリミティブ面で構成されるオブジェクトの描画（レンダリング）処理を高速に実行するものである。オブジェクトの描画の際には、メインプロセッサ900は、DMAコントローラ970の機能を利用して、オブジェクトデータを描画プロセッサ910に渡すと共に、必要であればテクスチャ記憶部924にテクスチャを転送する。すると、描画プロセッサ910は、これらのオブジェクトデータやテクスチャに基づいて、Zバッファなどを利用した陰面消去を行いながら、オブジェクトをフレームバッファ922に高速に描画する。また、描画プロセッサ910は、 $\alpha$ ブレンディング（半透明処理）、デプスキューイング、ミップマッピング、フォグ処理、バイリニア・フィルタリング、トライリニア・フィルタリング、アンチエイリアシング、シェーディング処理なども行うことができる。そして、1フレーム分の画像がフレームバッファ922に書き込まれると、その画像はディスプレイ912に表示される。

【0186】

サウンドプロセッサ930は、多チャンネルのADPCM音源などを内蔵し、BGM、効果音、音声などの高品位のゲーム音を生成する。生成されたゲーム音は、スピーカ932から出力される。

【0187】

ゲームコントローラ942からの操作データや、メモ리카ード944からのセ

ープデータ、個人データは、シリアルインターフェース940を介してデータ転送される。

---

【0188】

ROM950にはシステムプログラムなどが格納される。なお、業務用ゲームシステムの場合には、ROM950が情報記憶媒体として機能し、ROM950に各種プログラムが格納されることになる。なお、ROM950の代わりにハードディスクを利用するようにしてもよい。

【0189】

RAM960は、各種プロセッサの作業領域として用いられる。

【0190】

DMAコントローラ970は、プロセッサ、メモリ(RAM、VRAM、ROM等)間でのDMA転送を制御するものである。

【0191】

CDドライブ980は、プログラム、画像データ、或いは音データなどが格納されるCD982(情報記憶媒体)を駆動し、これらのプログラム、データへのアクセスを可能にする。

【0192】

通信インターフェース990は、ネットワークを介して外部との間でデータ転送を行うためのインターフェースである。この場合に、通信インターフェース990に接続されるネットワークとしては、通信回線(アナログ電話回線、ISDN)、高速シリアルバスなどを考えることができる。そして、通信回線を利用することでインターネットを介したデータ転送が可能になる。また、高速シリアルバスを利用することで、他のゲームシステムとの間でのデータ転送が可能になる。

【0193】

なお、本発明の各手段は、その全てを、ハードウェアのみにより実行してもよいし、情報記憶媒体に格納されるプログラムや通信インターフェースを介して配信されるプログラムのみにより実行してもよい。或いは、ハードウェアとプログラムの両方により実行してもよい。

## 【0194】

そして、本発明の各手段をハードウェアとプログラムの両方により実行する場合には、情報記憶媒体には、本発明の各手段をハードウェアを利用して実行するためのプログラムが格納されることになる。より具体的には、上記プログラムが、ハードウェアである各プロセッサ902、904、906、910、930等に処理を指示すると共に、必要であればデータを渡す。そして、各プロセッサ902、904、906、910、930等は、その指示と渡されたデータとに基づいて、本発明の各手段を実行することになる。

## 【0195】

図30(A)に、本実施形態を業務用ゲームシステムに適用した場合の例を示す。プレーヤは、ディスプレイ1100上に映し出されたゲーム画像を見ながら、レバー1102、ボタン1104等进行操作してゲームを楽しむ。内蔵されるシステムボード(サーキットボード)1106には、各種プロセッサ、各種メモリなどが実装される。そして、本発明の各手段を実行するための情報(プログラム又はデータ)は、システムボード1106上の情報記憶媒体であるメモリ1108に格納される。以下、この情報を格納情報と呼ぶ。

## 【0196】

図30(B)に、本実施形態を家庭用のゲームシステムに適用した場合の例を示す。プレーヤはディスプレイ1200に映し出されたゲーム画像を見ながら、ゲームコントローラ1202、1204进行操作してゲームを楽しむ。この場合、上記格納情報は、本体システムに着脱自在な情報記憶媒体であるCD1206、或いはメモ리카ード1208、1209等に格納されている。

## 【0197】

図30(C)に、ホスト装置1300と、このホスト装置1300とネットワーク1302(LANのような小規模ネットワークや、インターネットのような広域ネットワーク)を介して接続される端末1304-1~1304-n(ゲーム装置、携帯型電話)とを含むシステムに本実施形態を適用した場合の例を示す。この場合、上記格納情報は、例えばホスト装置1300が制御可能な磁気ディスク装置、磁気テープ装置、メモリ等の情報記憶媒体1306に格納されている。端

末 1 3 0 4 -1 ~ 1 3 0 4 -n が、スタンドアロンでゲーム画像、ゲーム音を生成できるものである場合には、ホスト装置 1 3 0 0 からは、ゲーム画像、ゲーム音を生成するためのゲームプログラム等が端末 1 3 0 4 -1 ~ 1 3 0 4 -n に配送される。一方、スタンドアロンで生成できない場合には、ホスト装置 1 3 0 0 がゲーム画像、ゲーム音を生成し、これを端末 1 3 0 4 -1 ~ 1 3 0 4 -n に伝送し端末において出力することになる。

## 【 0 1 9 8 】

なお、図 3 0 (C) の構成の場合に、本発明の各手段を、ホスト装置（サーバー）と端末とで分散して実行するようにしてもよい。また、本発明の各手段を実行するための上記格納情報を、ホスト装置（サーバー）の情報記憶媒体と端末の情報記憶媒体に分散して格納するようにしてもよい。

## 【 0 1 9 9 】

またネットワークに接続する端末は、家庭用ゲームシステムであってもよいし業務用ゲームシステムであってもよい。そして、業務用ゲームシステムをネットワークに接続する場合には、業務用ゲームシステムとの間で情報のやり取りが可能であると共に家庭用ゲームシステムとの間でも情報のやり取りが可能な携帯型情報記憶装置（メモリカード、携帯型ゲーム装置）を用いることが望ましい。

## 【 0 2 0 0 】

なお本発明は、上記実施形態で説明したものに限らず、種々の変形実施が可能である。

## 【 0 2 0 1 】

例えば、本発明のうち従属請求項に係る発明においては、従属先の請求項の構成要件の一部を省略する構成とすることもできる。また、本発明の 1 の独立請求項に係る発明の要部を、他の独立請求項に従属させることもできる。

## 【 0 2 0 2 】

また、奥行き値の第 2 の奥行き値への変換は、図 1 7 で説明したようにインデックスカラー・テクスチャマッピング用のルックアップテーブルを利用する手法で実現することが特に望ましいが、他の手法により実現してもよい。

## 【 0 2 0 3 】

またルックアップテーブルの変換特性も図18、図19、図20(A)、(B)に示した変換特性に限定されず、種々の変形実施が可能である。

---

**【0204】**

また、元画像の合成対象となるぼかし画像は、図23、図24で説明した手法により生成することが特に望ましいが、これに限定されない。例えば元画像と元画像をずらした画像を合成したり、当該フレームの元画像と前のフレームの元画像とを合成したりしてぼかし画像を生成してもよい。

**【0205】**

また、テクセル補間方式を利用してぼかし画像を生成する発明も、図23～図26(B)で説明した手法に限定されない。例えば、画面全体をぼかすのではなく、画面よりも小さいぼかし領域を設定して、その領域にある元画像をぼかすようにしてもよい。

**【0206】**

また本実施形態では視点から近いほど奥行き値が大きくなる場合を例にとり説明したが、視点から遠いほど奥行き値が大きくなる場合にも本発明は適用できる。

**【0207】**

また、本発明は種々のゲーム（格闘ゲーム、シューティングゲーム、ロボット対戦ゲーム、スポーツゲーム、競争ゲーム、ロールプレイングゲーム、音楽演奏ゲーム、ダンスゲーム等）に適用できる。

**【0208】**

また本発明は、業務用ゲームシステム、家庭用ゲームシステム、多数のプレーヤが参加する大型アトラクションシステム、シミュレータ、マルチメディア端末、ゲーム画像を生成するシステムボード等の種々のゲームシステム（画像生成システム）に適用できる。

**【図面の簡単な説明】****【図1】**

本実施形態のゲームシステムのブロック図の例である。

**【図2】**

図 2 (A)、(B) は、ガンマ補正を実現する第 1、第 2 の手法について説明するための図である。

---

【図 3】

インデックスカラー・テクスチャマッピングについて説明するための図である。

【図 4】

インデックスカラー・テクスチャマッピング用の LUT を有効利用して、元画像を変換する手法について説明するための図である。

【図 5】

図 5 (A)、(B) は、本実施形態により生成されるゲーム画像の例である。

【図 6】

元画像を複数のブロックに分割し、各ブロックの画像を、LUT を用いて分割ブロックサイズのポリゴンにテクスチャマッピングする手法について説明するための図である。

【図 7】

LUT の変換により得られた画像情報のうち、必要な画像情報のみが描画領域に描画され、他の画像情報が描画されないようにマスク処理を行う手法について説明するための図である。

---

【図 8】

LUT を利用したテクスチャマッピングを行い、 $\alpha$  プレーンを作成する手法について説明するための図である。

【図 9】

Z 値を LUT のインデックス番号に設定する手法について説明するための図である。

【図 10】

Z 値に応じた  $\alpha$  値を設定し、設定された  $\alpha$  値を用いて元画像とぼかし画像を合成する手法について説明するための図である。

【図 11】

図 11 (A)、(B) は、元画像とそのぼかし画像の例である。

## 【図 1 2】

Z 値に応じた  $\alpha$  値の設定手法について説明するための図である。

## 【図 1 3】

図 1 3 (A)、(B)、(C) は、仮想オブジェクトを描画することで仮想オブジェクトの奥側の画素の  $\alpha$  値を更新する手法について説明するための図である。

## 【図 1 4】

Z 値を Z 2 値に変換し、その Z 2 値を  $\alpha$  値に変換して元画像とぼかし画像を合成する手法について説明するための図である。

## 【図 1 5】

Z 値の最上位ビットを含む上位ビットで Z 2 値を構成した場合の問題点について説明するための図である。

## 【図 1 6】

Z 値の最上位ビットよりも下位のビット I ~ J ビットで Z 2 値を構成すると共に Z 2 値を所与の値にクランプする手法について説明するための図である。

## 【図 1 7】

LUT を用いて Z 値を Z 2 値に変換する手法について説明するための図である。

## 【図 1 8】

Z 値のビット 1 5 ~ 8 を変換する LUT 1 の具体例を示す図である。

## 【図 1 9】

Z 値のビット 2 3 ~ 1 6 を変換する LUT 2 の具体例を示す図である。

## 【図 2 0】

図 2 0 (A)、(B) は、Z 2 値を  $\alpha$  値に変換する LUT 3 の具体例とその変換特性曲線の例を示す図である。

## 【図 2 1】

クランプ処理について説明するための図である。

## 【図 2 2】

バイリニアフィルタ方式のテクスチャマッピングについて説明するための図で

ある。

【図 2 3】

バイリニアフィルタ方式を有効利用してぼかし画像を生成する手法について説明するための図である。

【図 2 4】

バイリニアフィルタ方式を有効利用してぼかし画像を生成する手法について説明するための図である。

【図 2 5】

図 2 5 (A)、(B) は、バイリニアフィルタ方式の補間機能によりぼかし画像が生成される原理について説明するための図である。

【図 2 6】

図 2 6 (A)、(B) も、バイリニアフィルタ方式の補間機能によりぼかし画像が生成される原理について説明するための図である。

【図 2 7】

本実施形態の処理の詳細例について示すフローチャートである。

【図 2 8】

本実施形態の処理の詳細例について示すフローチャートである。

【図 2 9】

本実施形態を実現できるハードウェアの構成の一例を示す図である。

【図 3 0】

図 3 0 (A)、(B)、(C) は、本実施形態が適用される種々の形態のシステムの例を示す図である。

【符号の説明】

- 1 0 仮想カメラ
- 1 0 0 処理部
- 1 1 0 ゲーム処理部
- 1 1 2 移動・動作演算部
- 1 3 0 画像生成部
- 1 3 2 ジオメトリ処理部



1 3 4 インデックス番号設定部

1 4 0 描画部

---

1 4 2 テクスチャマッピング部

1 4 4  $\alpha$  合成部

1 4 6 陰面消去部

1 5 0 音生成部

1 6 0 操作部

1 7 0 記憶部

1 7 2 主記憶領域

1 7 4 フレームバッファ

1 7 6 テクスチャ記憶部

1 7 8 LUT 記憶部

1 7 9 Z バッファ

1 8 0 情報記憶媒体

1 9 0 表示部

1 9 2 音出力部

1 9 4 携帯型情報記憶装置

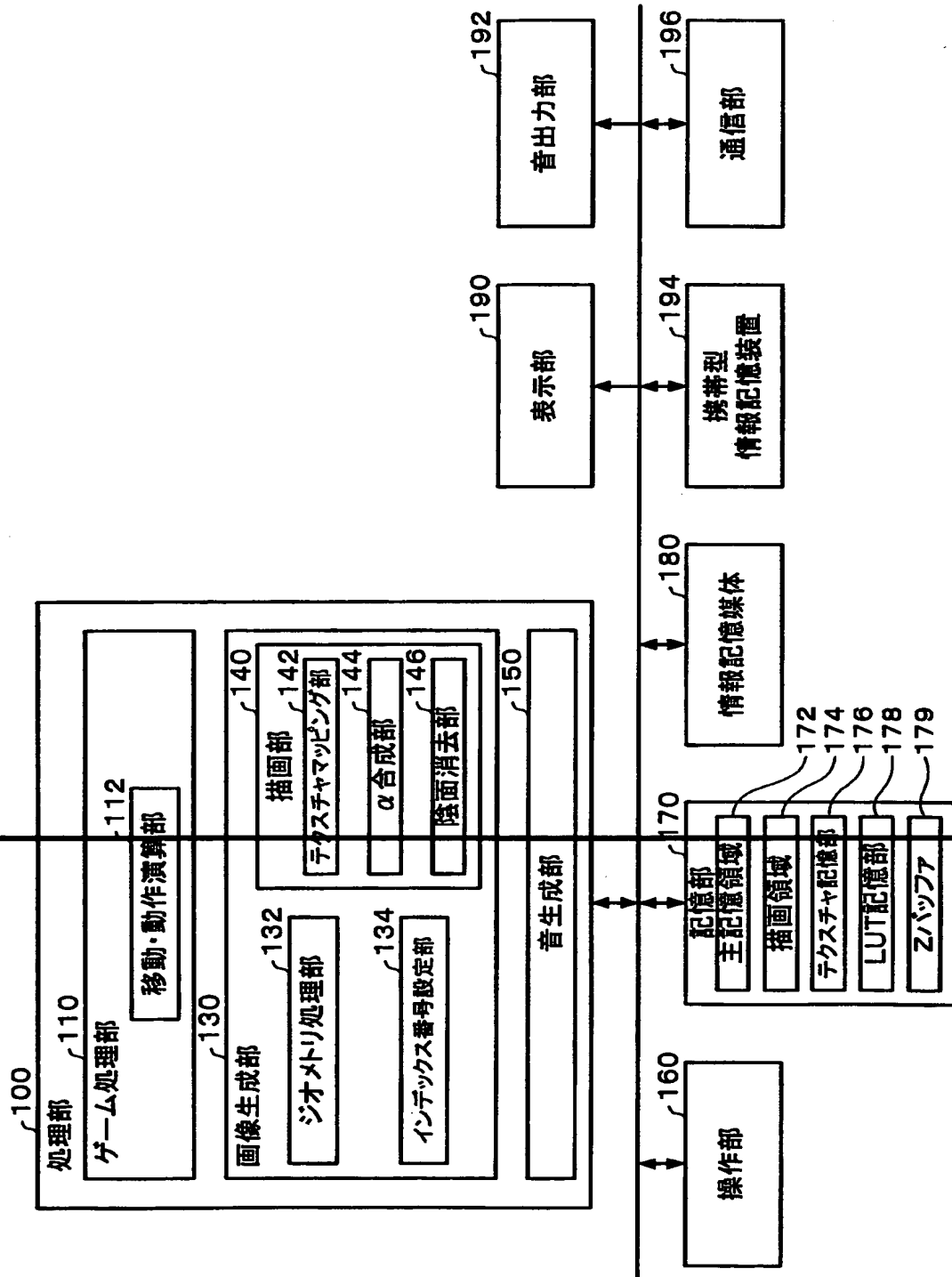
1 9 6 通信部

---

【書類名】

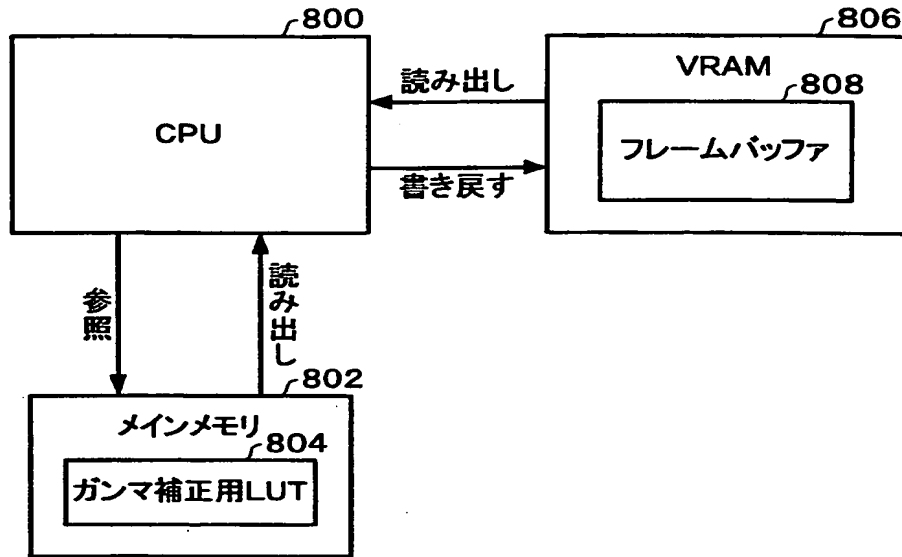
図面

【図 1】

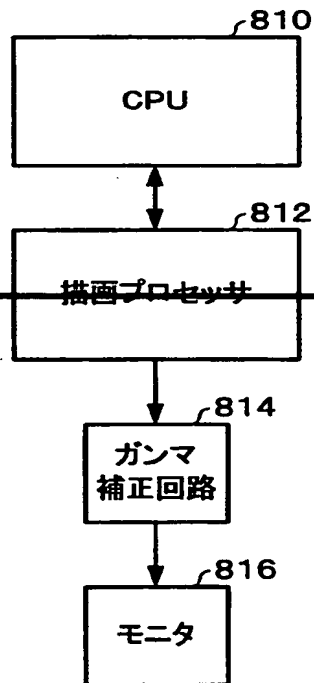


【図 2】

(A)

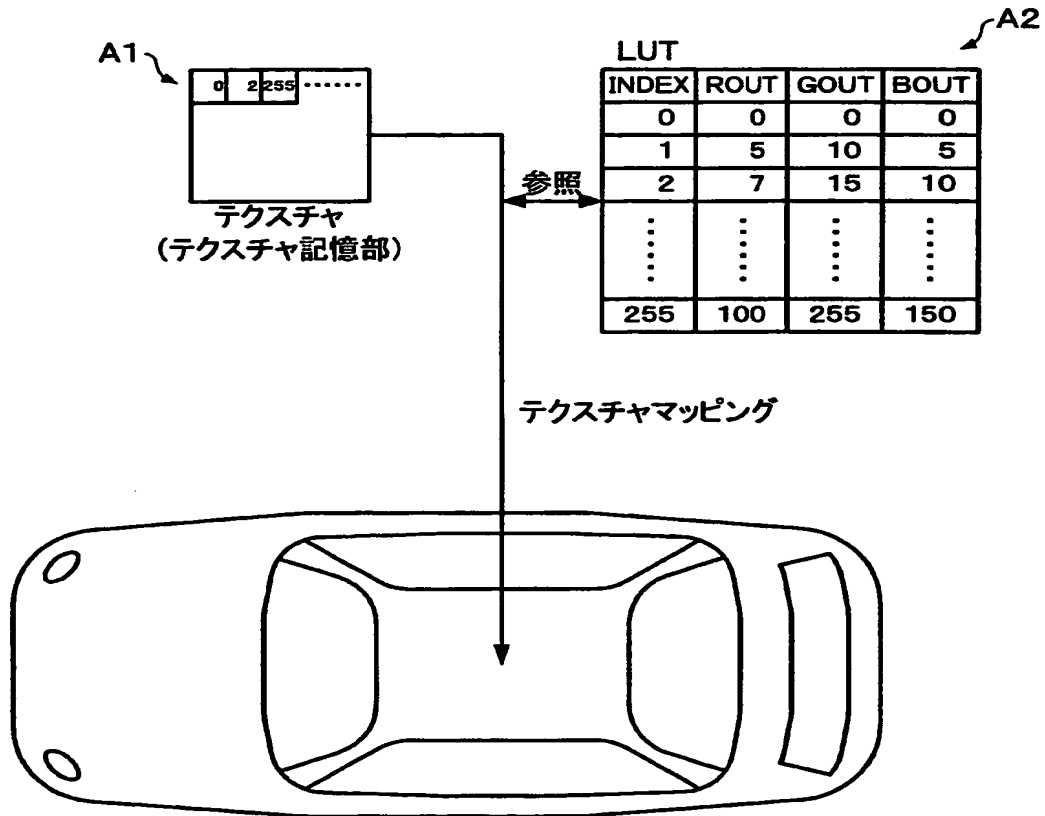


(B)

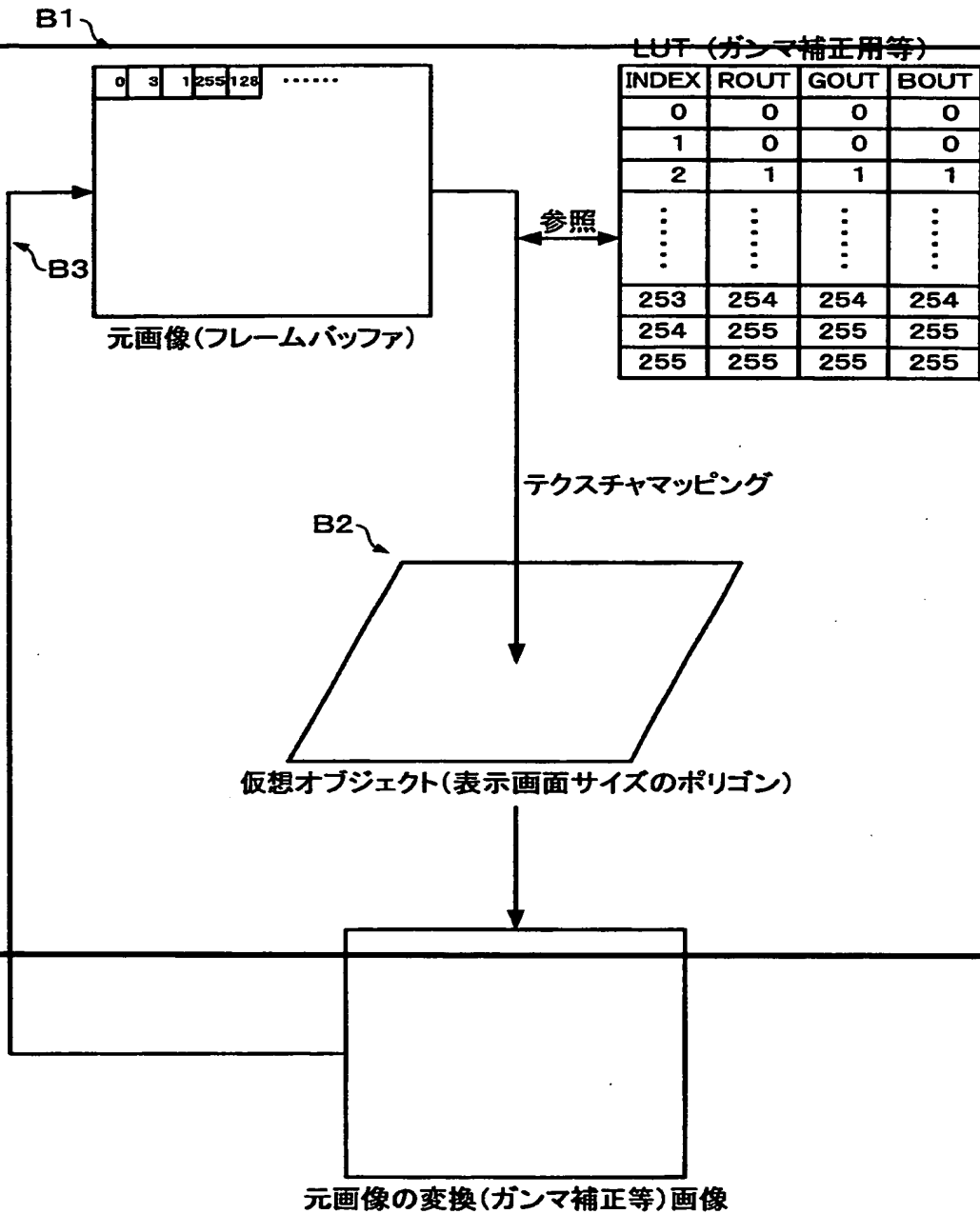


【図 3】

インデックスカラー・テクスチャマッピング

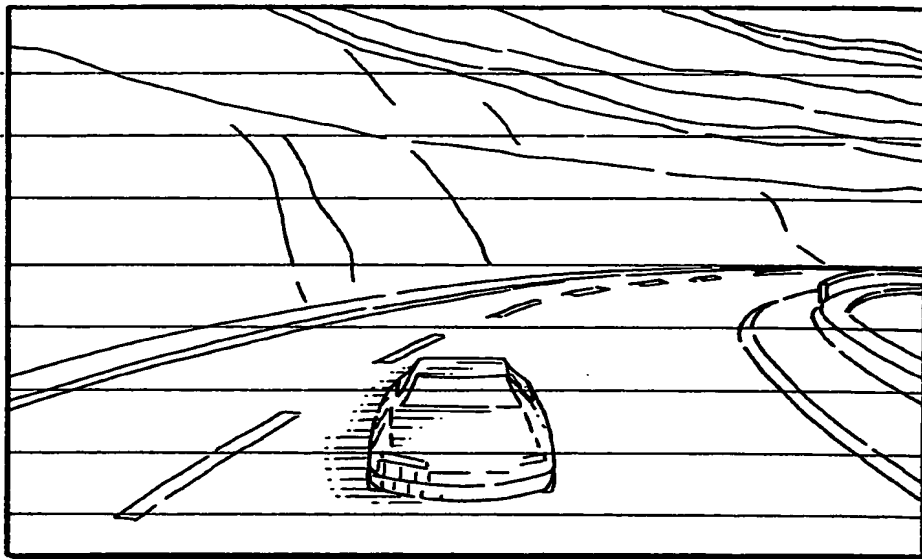


【図 4】

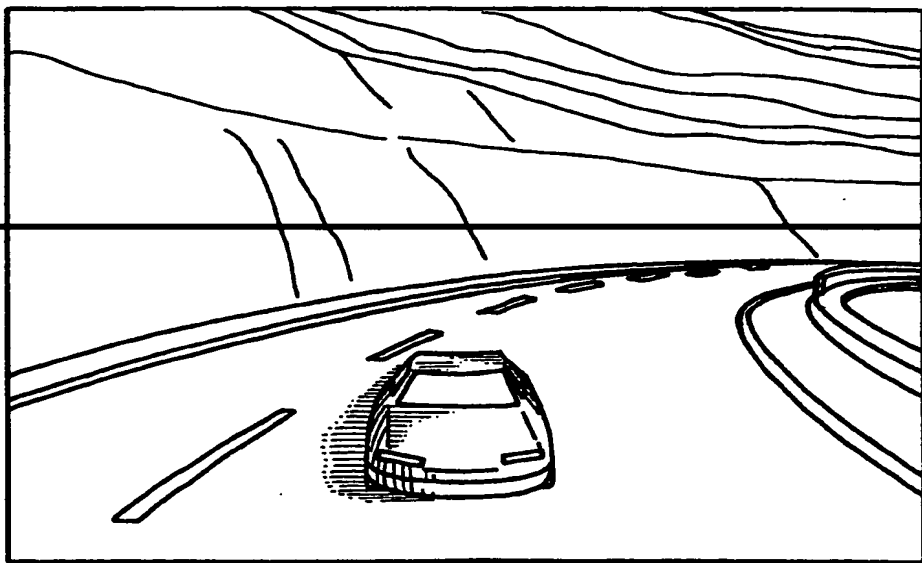


【図5】

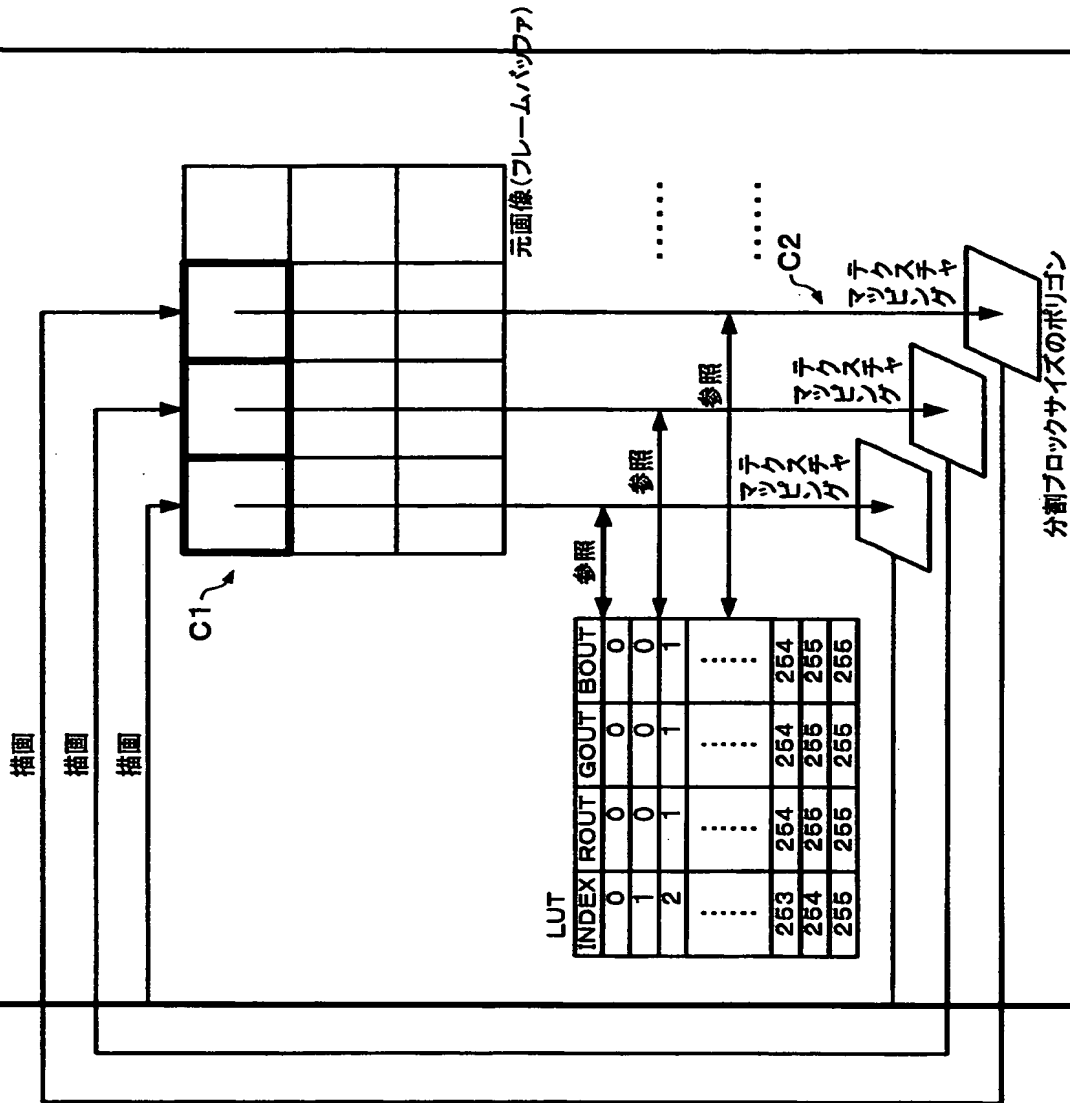
(A)



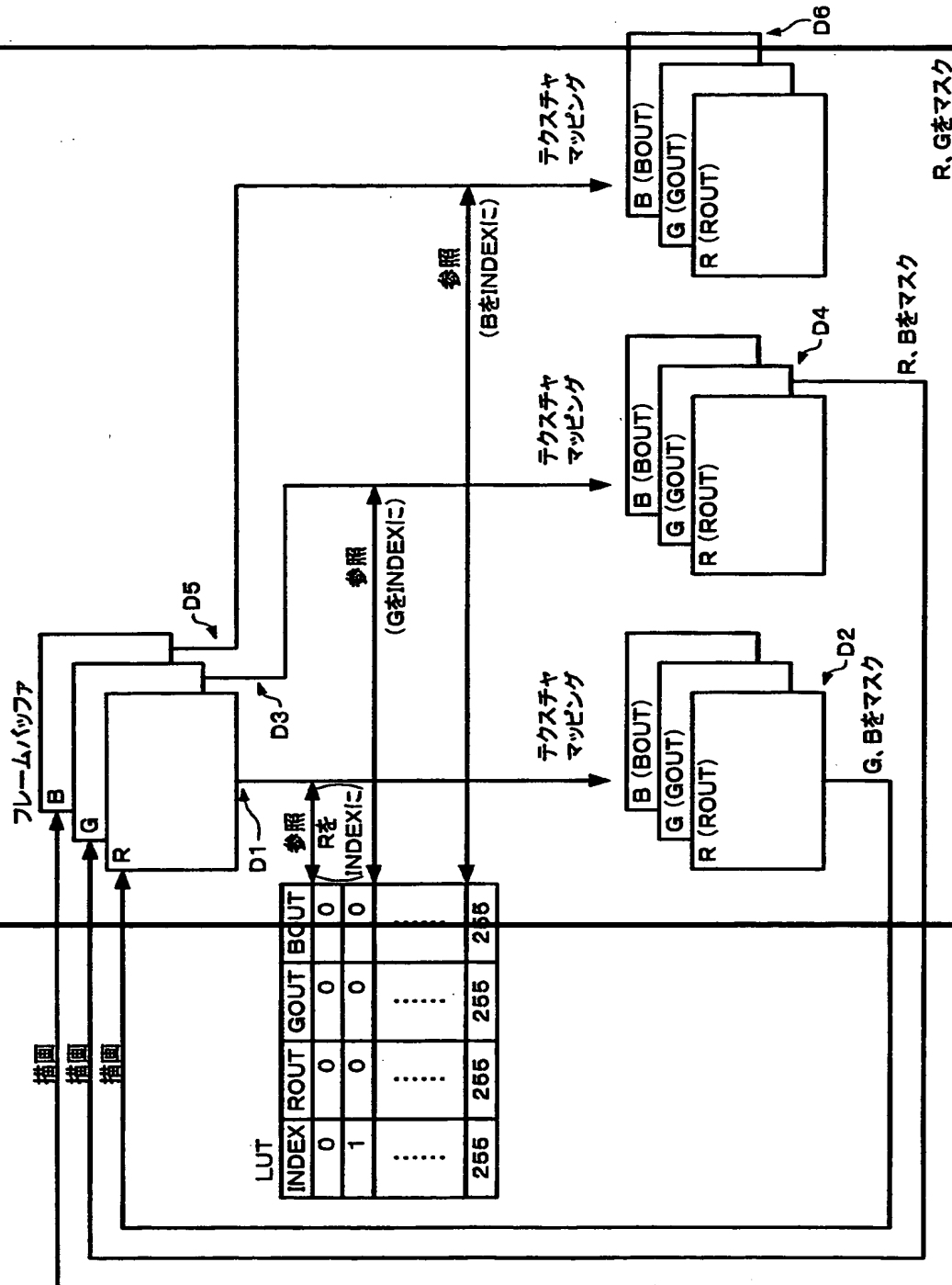
(B)



【図 6】



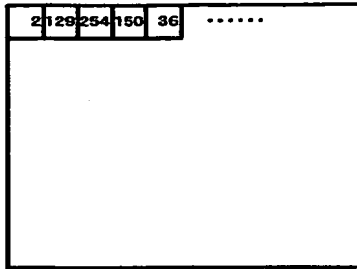
【図 7】





【図 8】

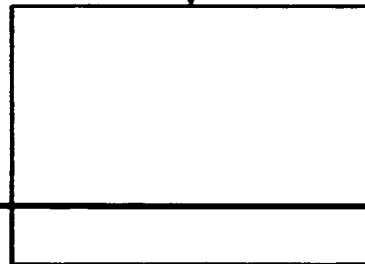
R(G, B)



LUT

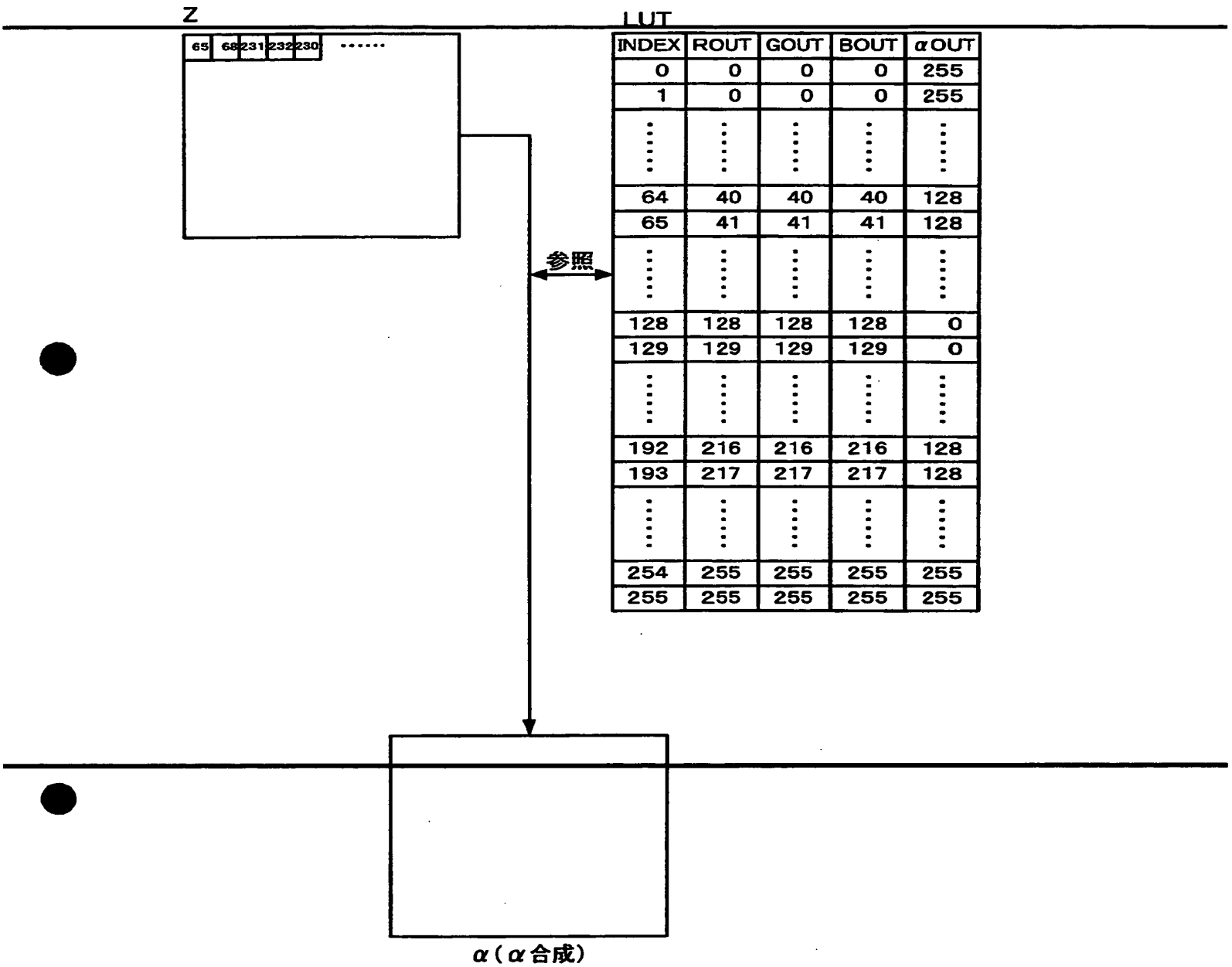
INDEX	ROUT	GOUT	BOUT	$\alpha$ OUT
0	0	0	0	0
1	0	0	0	0
2	1	1	1	0
⋮	⋮	⋮	⋮	⋮
127	127	127	127	0
128	128	128	128	255
129	129	129	129	255
⋮	⋮	⋮	⋮	⋮
253	254	254	254	255
254	255	255	255	255
255	255	255	255	255

参照

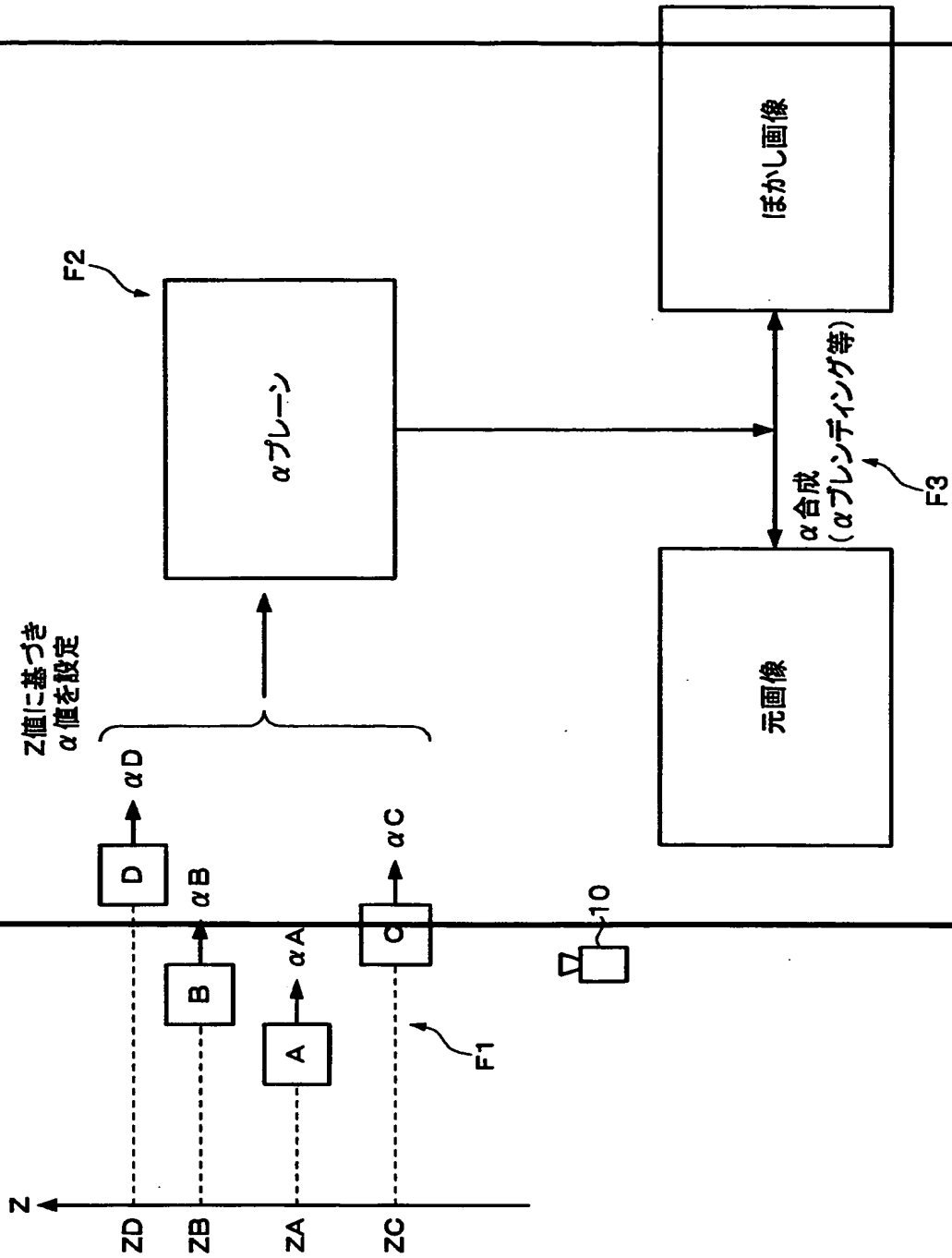


$\alpha$  (マスク)

【図 9】

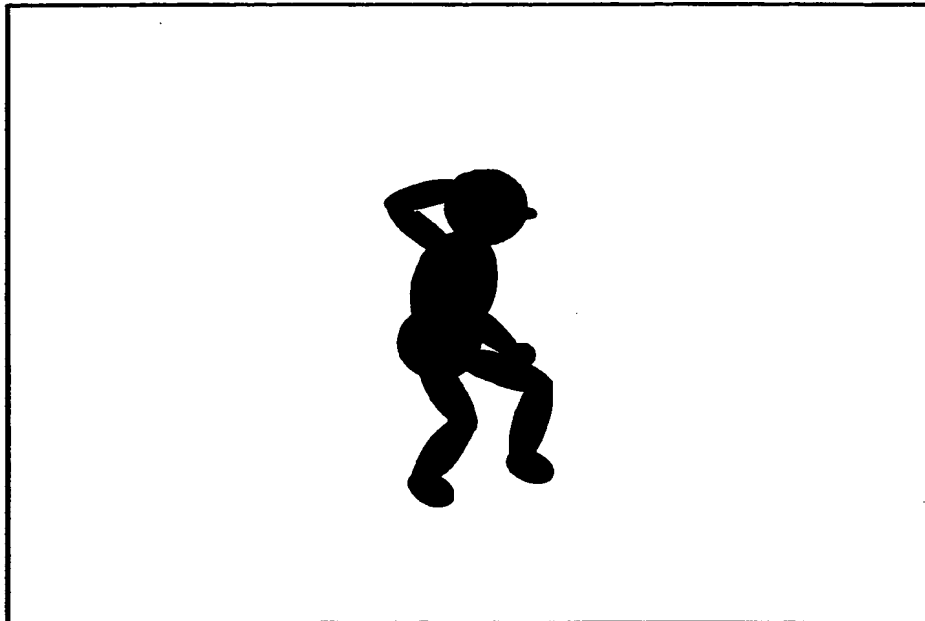


【図 1 0】

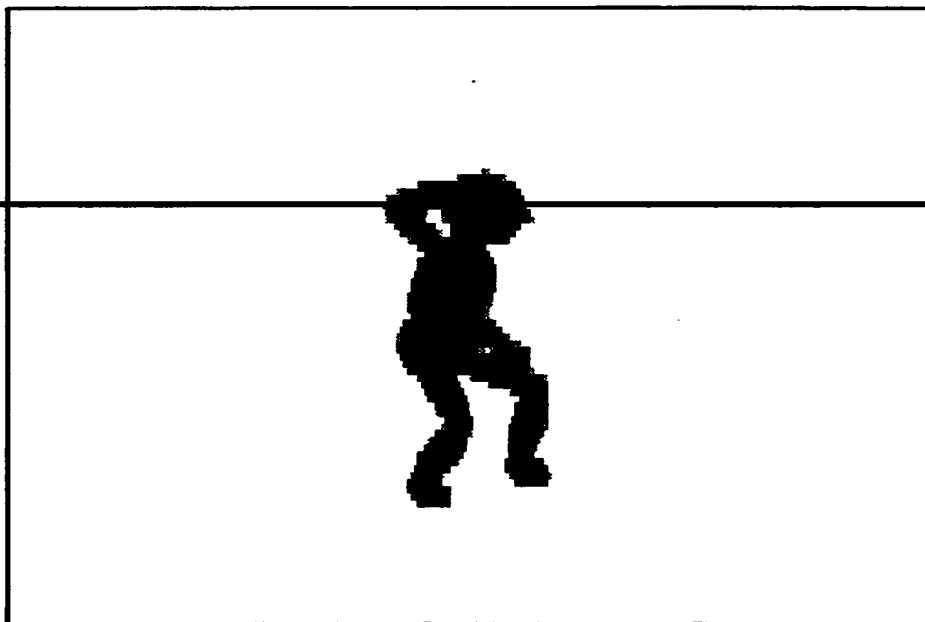


【図 1 1】

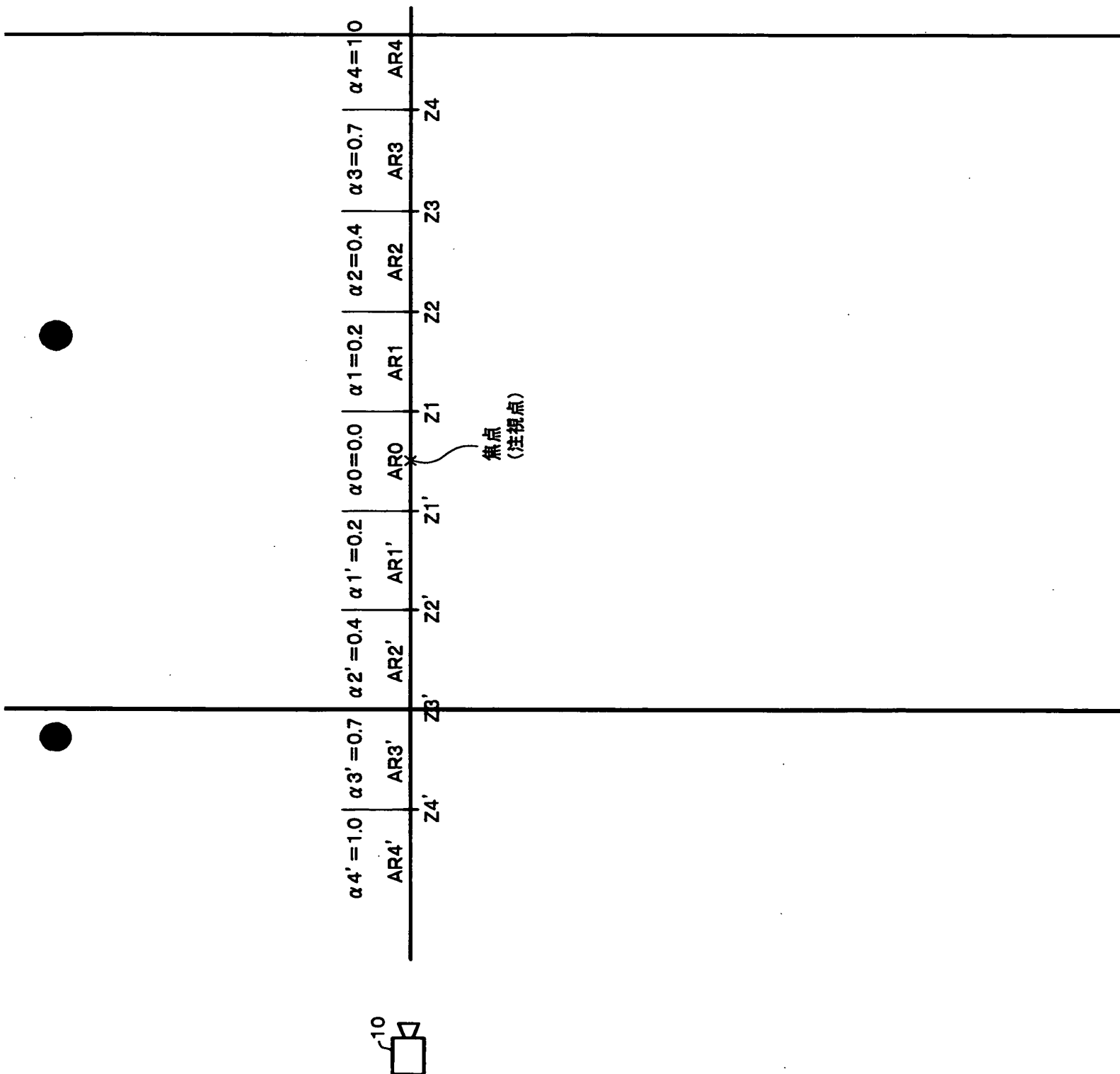
(A) 元画像



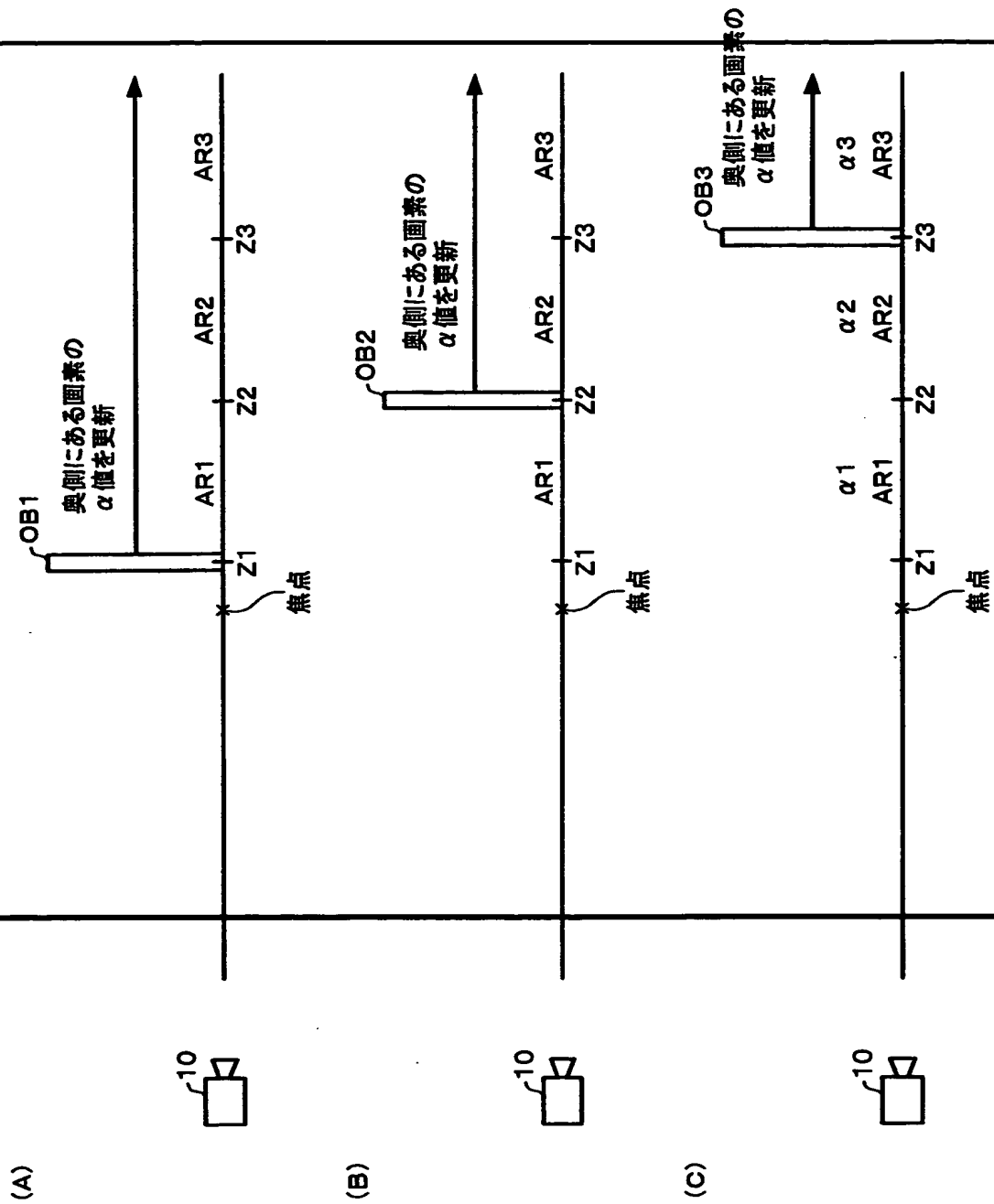
(B) ぼかし画像



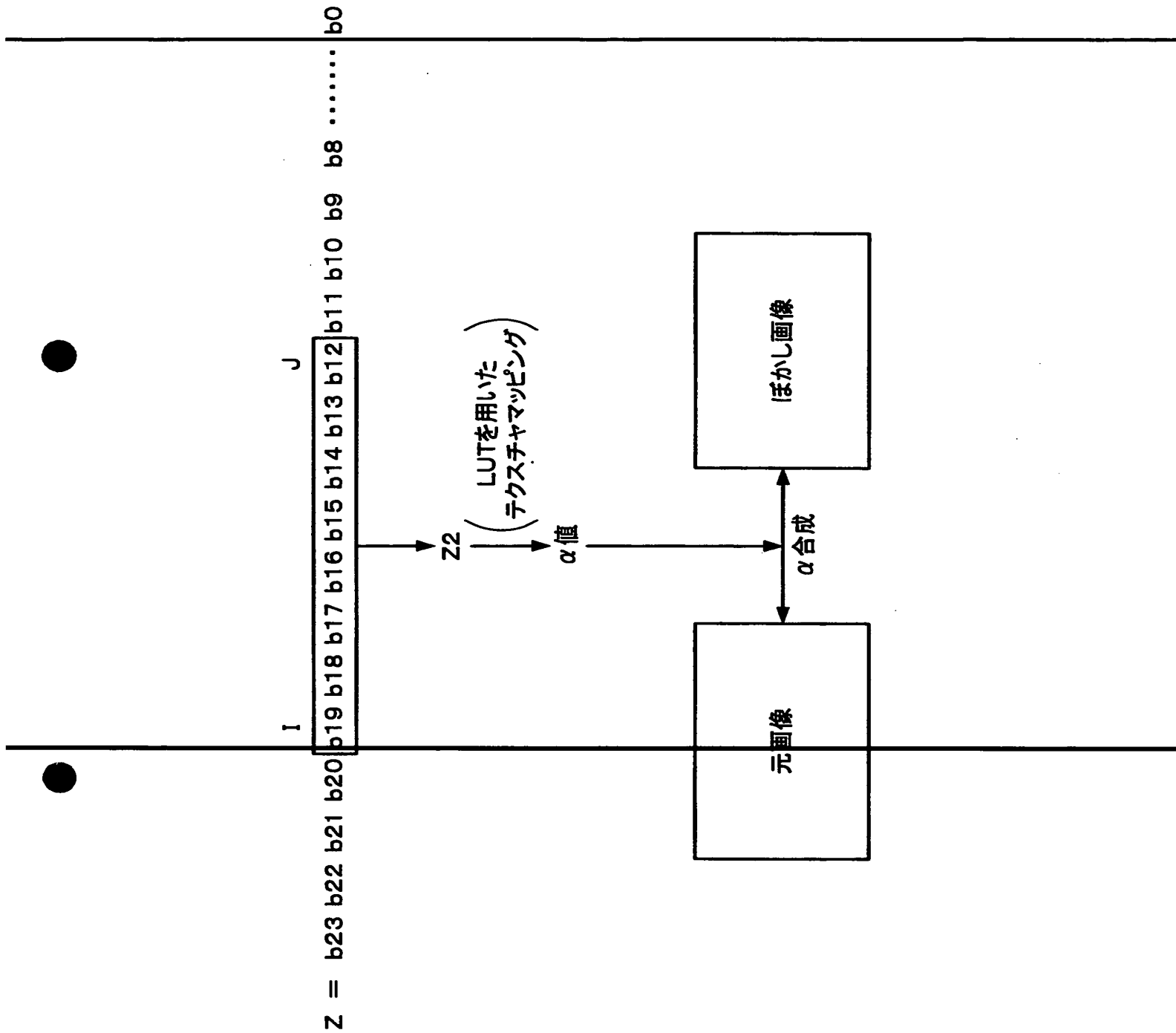
【図 1 2】



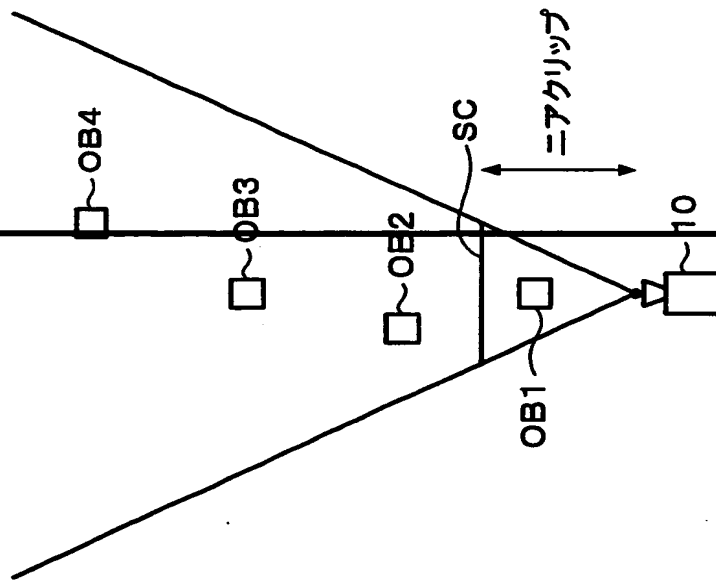
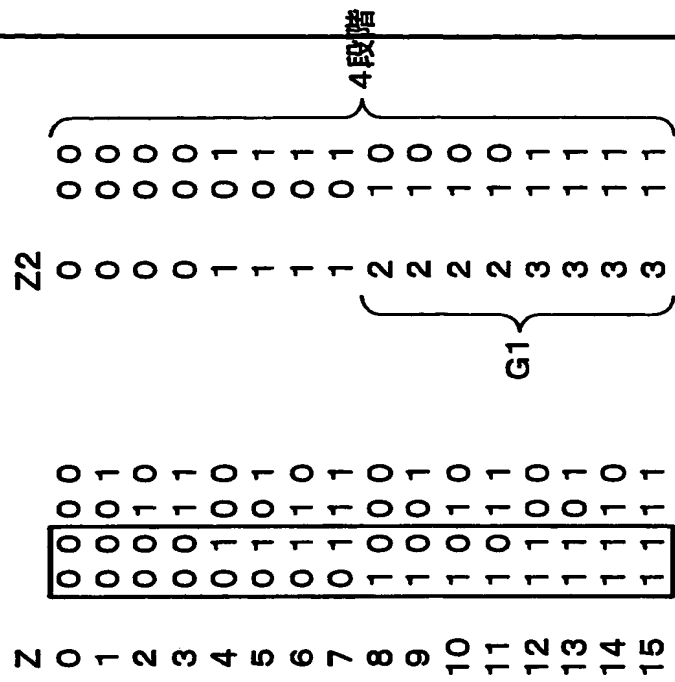
【図 13】



【図 14】

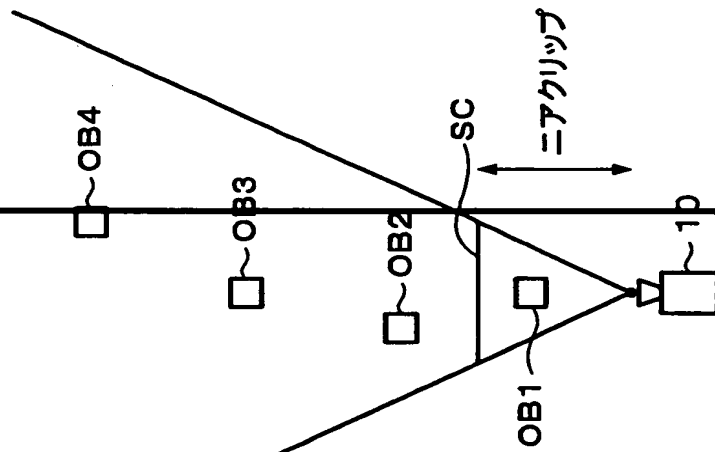
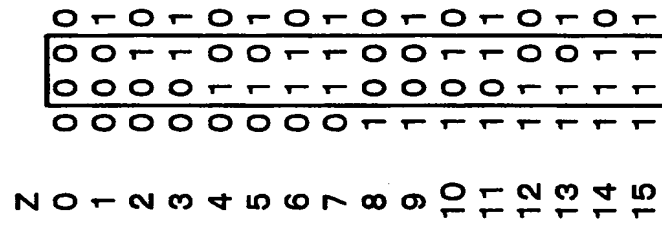
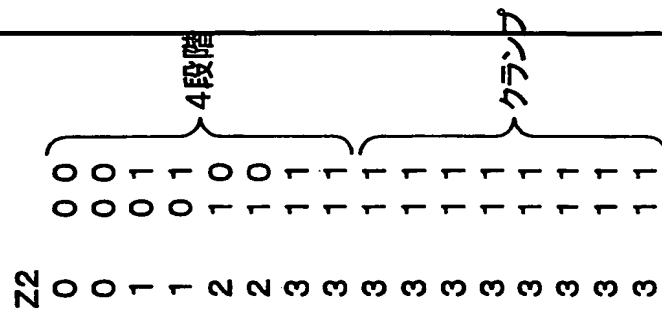


【図 15】

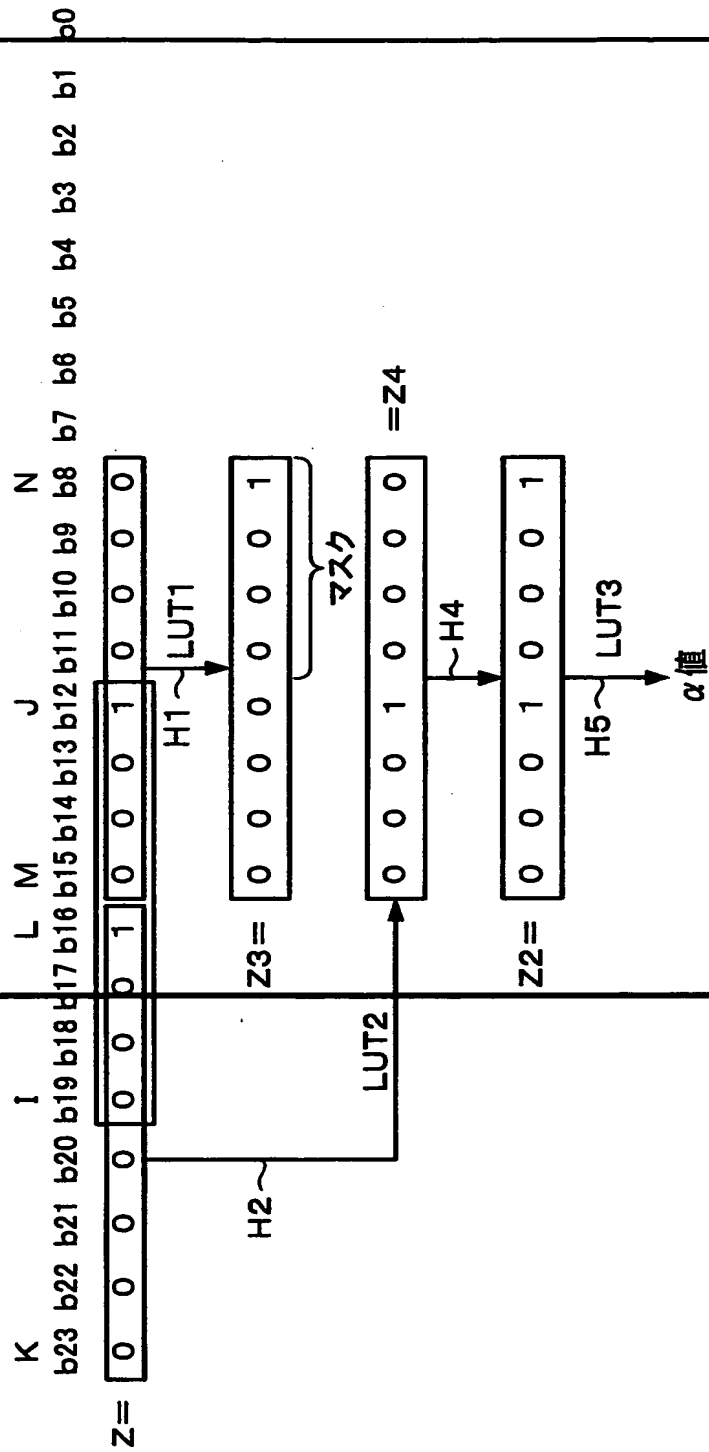




【図 16】



【図 17】



【図 1 8】

LUT1(ビット15～8)

INDEX	OUT (R,G,B, $\alpha$ のいずれか)
0x00 (00000000)	0x00 (00000000)
⋮	⋮
0x0F (00001111)	0x00 (00000000)
0x10 (00010000)	0x01 (00000001)
⋮	⋮
0x1F (00011111)	0x01 (00000001)
0x20 (00100000)	0x02 (00000010)
⋮	⋮
0x2F (00101111)	0x02 (00000010)
0x30 (00110000)	0x03 (00000011)
⋮	⋮
0xE0 (11100000)	0x0E (00001110)
⋮	⋮
0xEF (11101111)	0x0E (00001110)
0xF0 (11110000)	0x0F (00001111)
0xF1 (11110001)	0x0F (00001111)
0xF2 (11110010)	0x0F (00001111)
⋮	⋮
0xFF (11111111)	0x0F (00001111)

【図 1 9】

LUT2(ビット23~16)

INDEX	OUT (R,G,B, $\alpha$ のいずれか)
0x00 (00000000)	0x00 (00000000)
0x01 (00000001)	0x10 (00010000)
0x02 (00000010)	0x20 (00100000)
0x03 (00000011)	0x30 (00110000)
0x04 (00000100)	0x40 (01000000)
⋮	⋮
0x0E (00001110)	0xE0 (11100000)
0x0F (00001111)	0xF0 (11110000)
0x10 (00010000)	0xF0 (11110000)
0x11 (00010001)	0xF0 (11110000)
⋮	⋮
0xFF (11111111)	0xF0 (11110000)

Q1  
クランプ

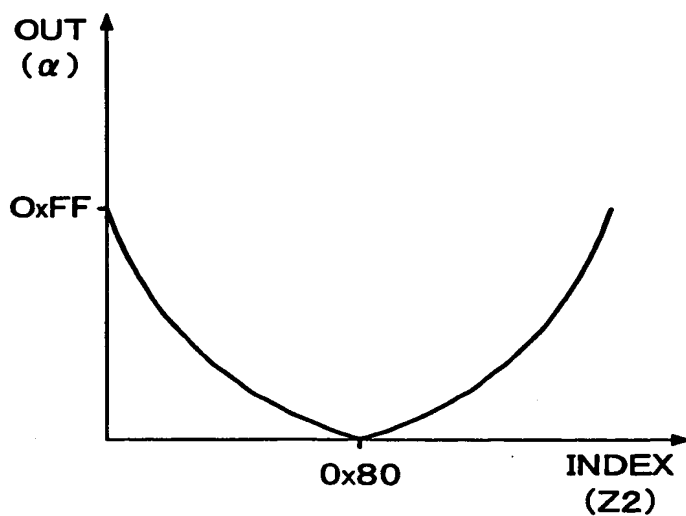
【図 2 0】

(A)

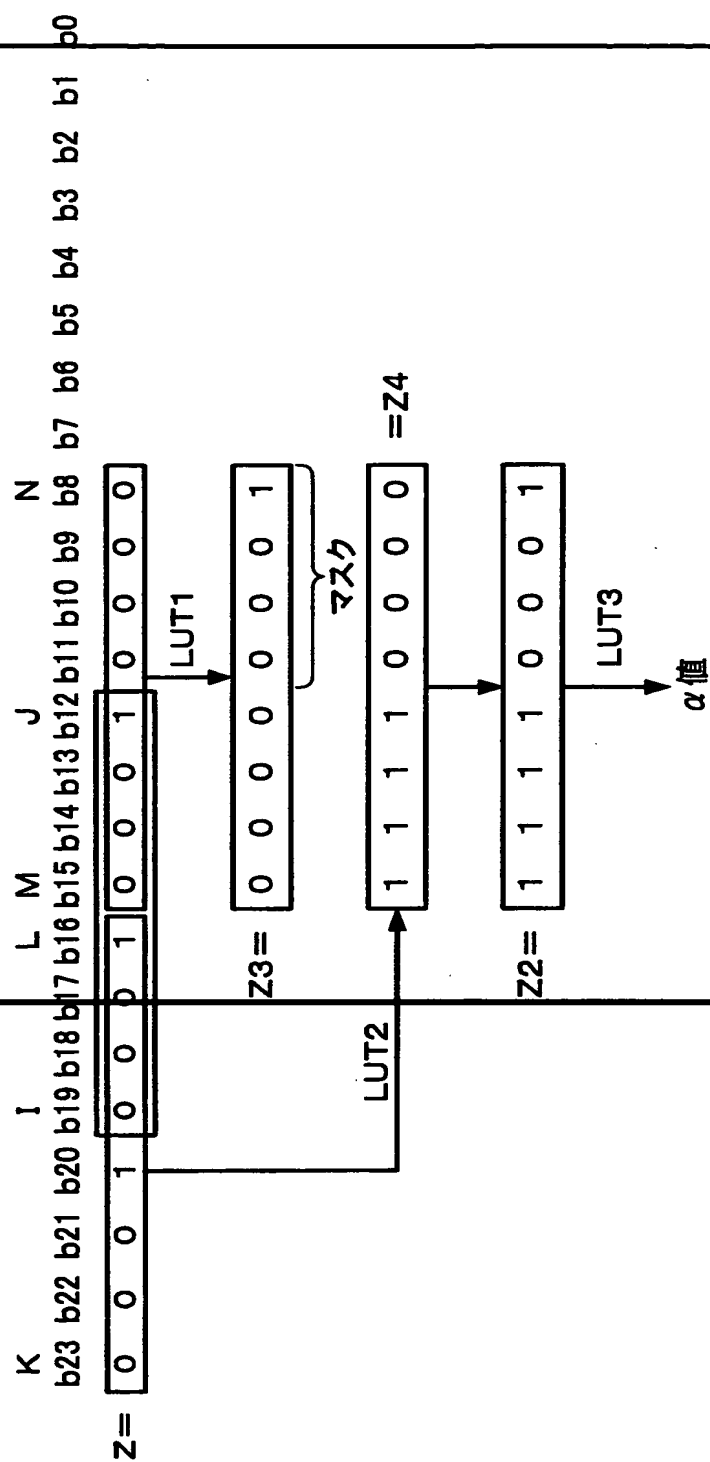
LUT3

INDEX	OUT( $\alpha$ )
0x00 (00000000)	0xFF (11111111)
0x01 (00000001)	0xFE (11111110)
0x02 (00000010)	0xFB (11111011)
⋮	⋮
0x7F (01111111)	0x00 (00000000)
0x80 (10000000)	0x00 (00000000)
0x81 (10000001)	0x00 (00000000)
⋮	⋮
0xFE (11111110)	0xFE (11111110)
0xFF (11111111)	0xFF (11111111)

(B)

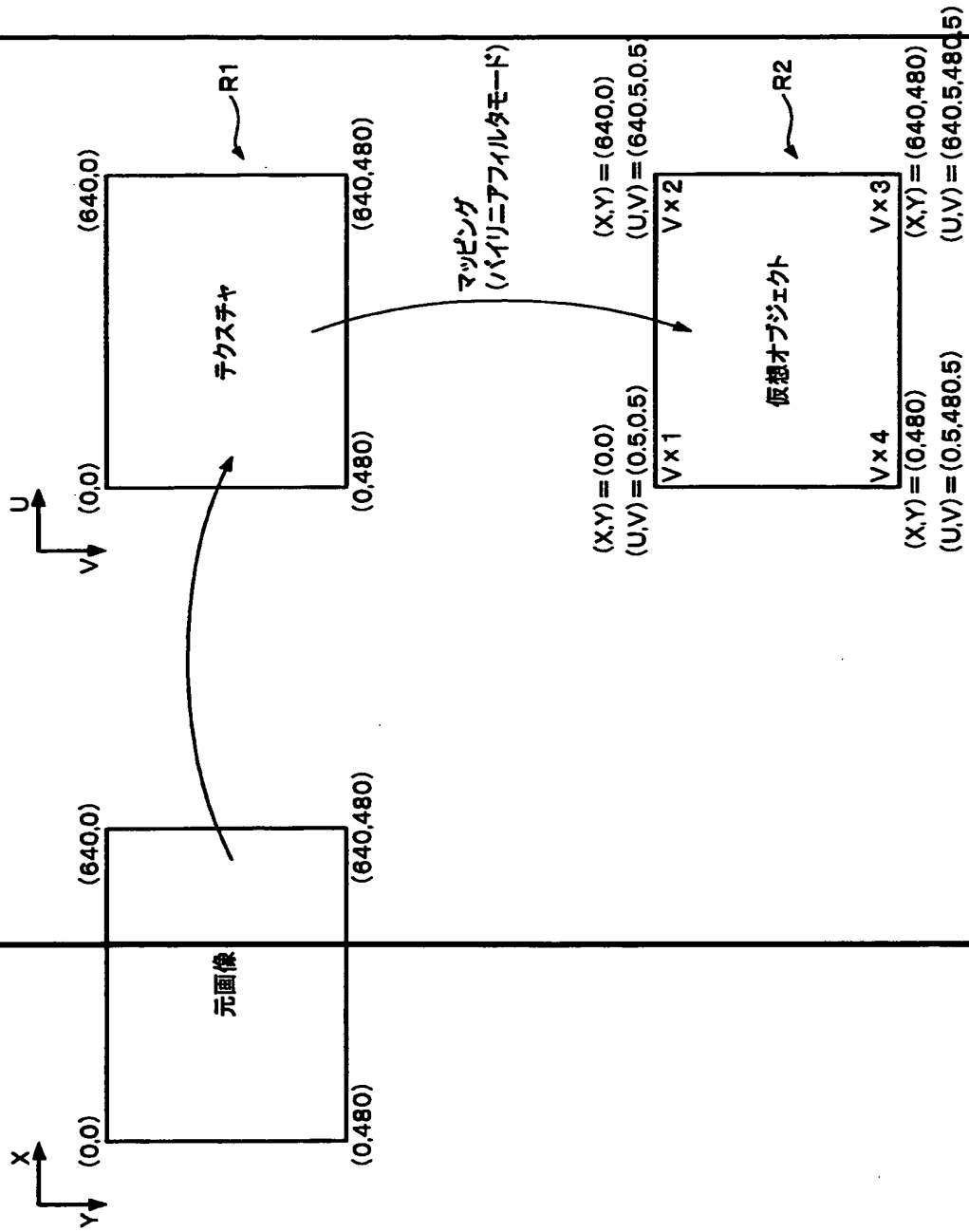


【図 21】



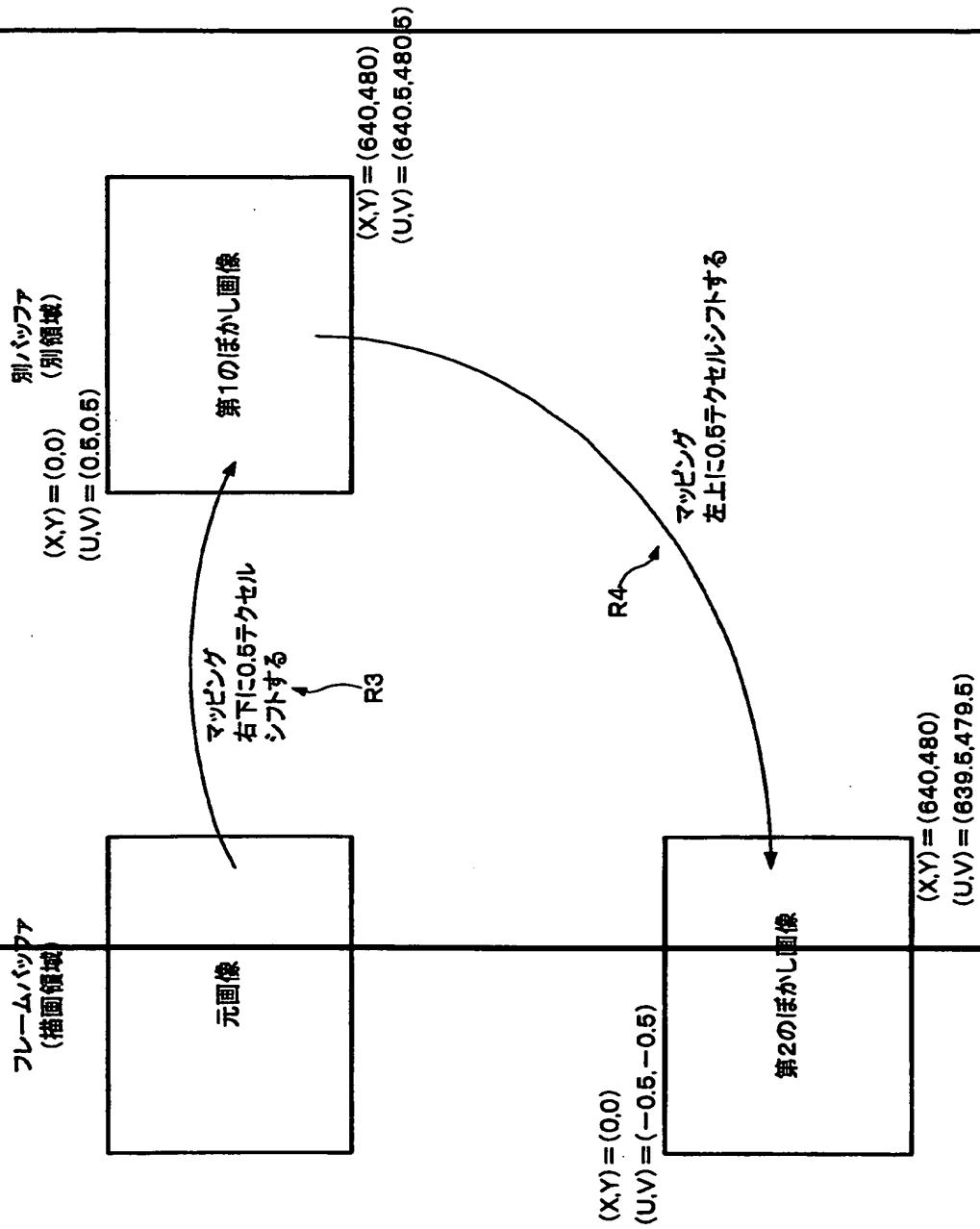


【図 2 3】



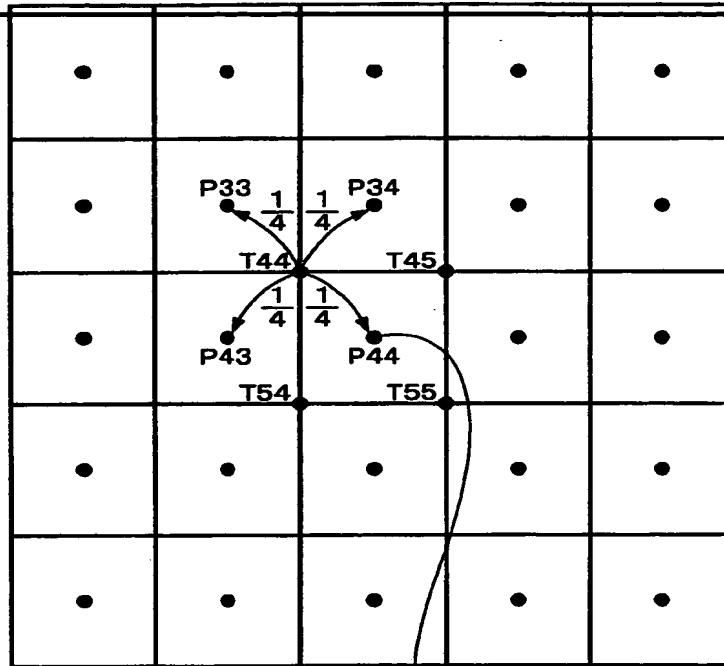


【図 2 4】



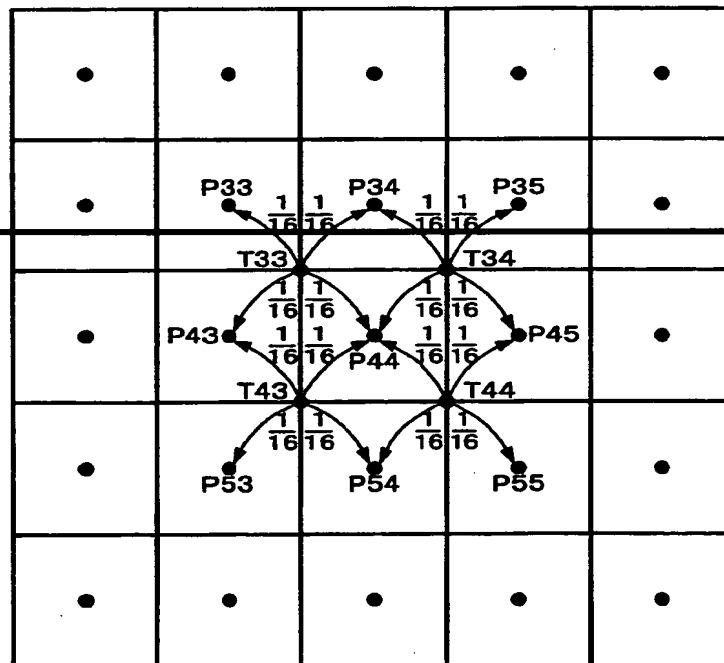
【図 25】

(A)



$$CP44 = (C44 + C45 + C54 + C55) / 4$$

(B)



【図 2 6】

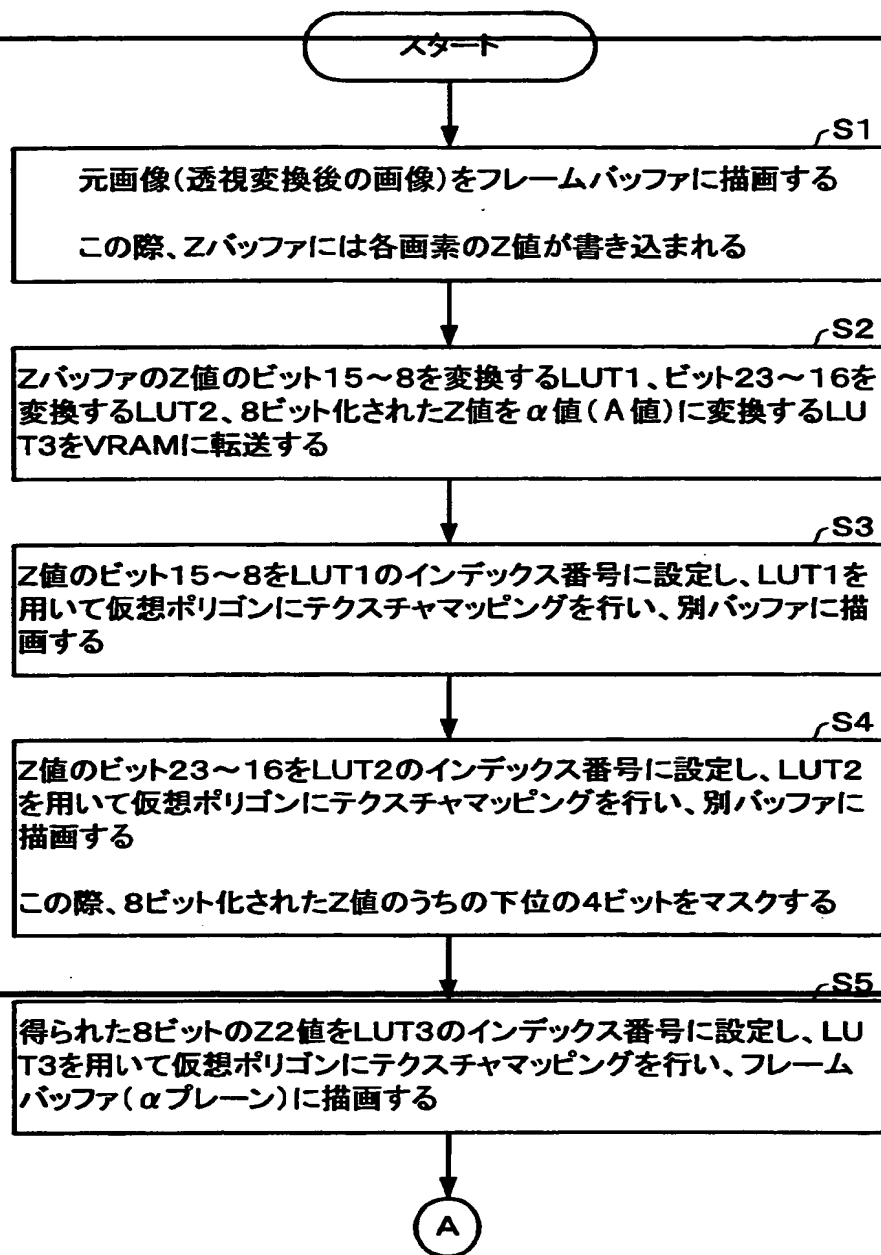
(A)

	$\frac{1}{16}$	$\frac{2}{16}$	$\frac{1}{16}$	
	$\frac{2}{16}$	$\frac{4}{16}$	$\frac{2}{16}$	
	$\frac{1}{16}$	$\frac{2}{16}$	$\frac{1}{16}$	

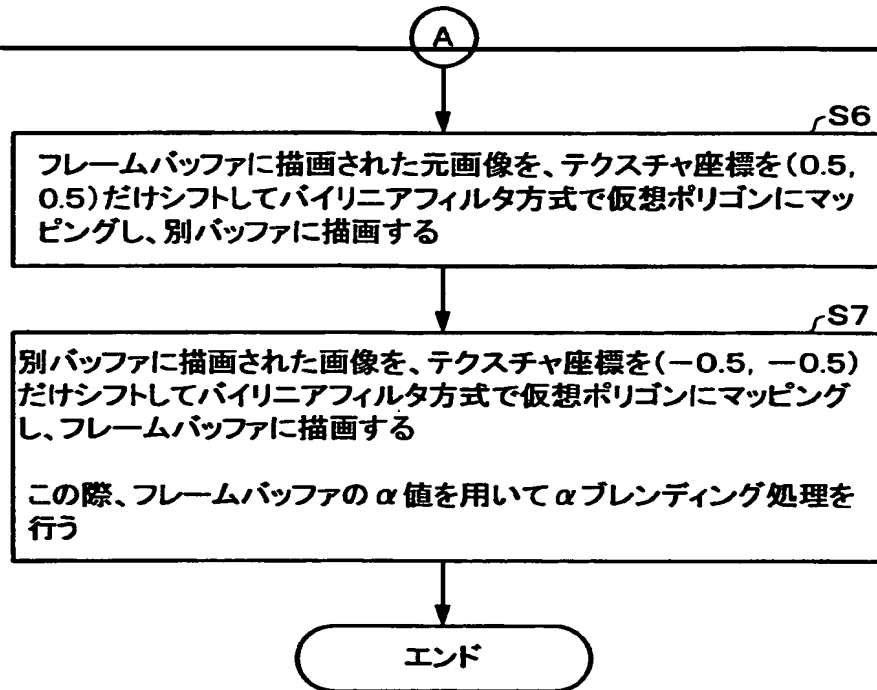
(B)

$\frac{1}{256}$	$\frac{4}{256}$	$\frac{6}{256}$	$\frac{4}{256}$	$\frac{1}{256}$
$\frac{4}{256}$	$\frac{16}{256}$	$\frac{24}{256}$	$\frac{16}{256}$	$\frac{4}{256}$
$\frac{6}{256}$	$\frac{24}{256}$	$\frac{36}{256}$	$\frac{24}{256}$	$\frac{6}{256}$
$\frac{4}{256}$	$\frac{16}{256}$	$\frac{24}{256}$	$\frac{16}{256}$	$\frac{4}{256}$
$\frac{1}{256}$	$\frac{4}{256}$	$\frac{6}{256}$	$\frac{4}{256}$	$\frac{1}{256}$

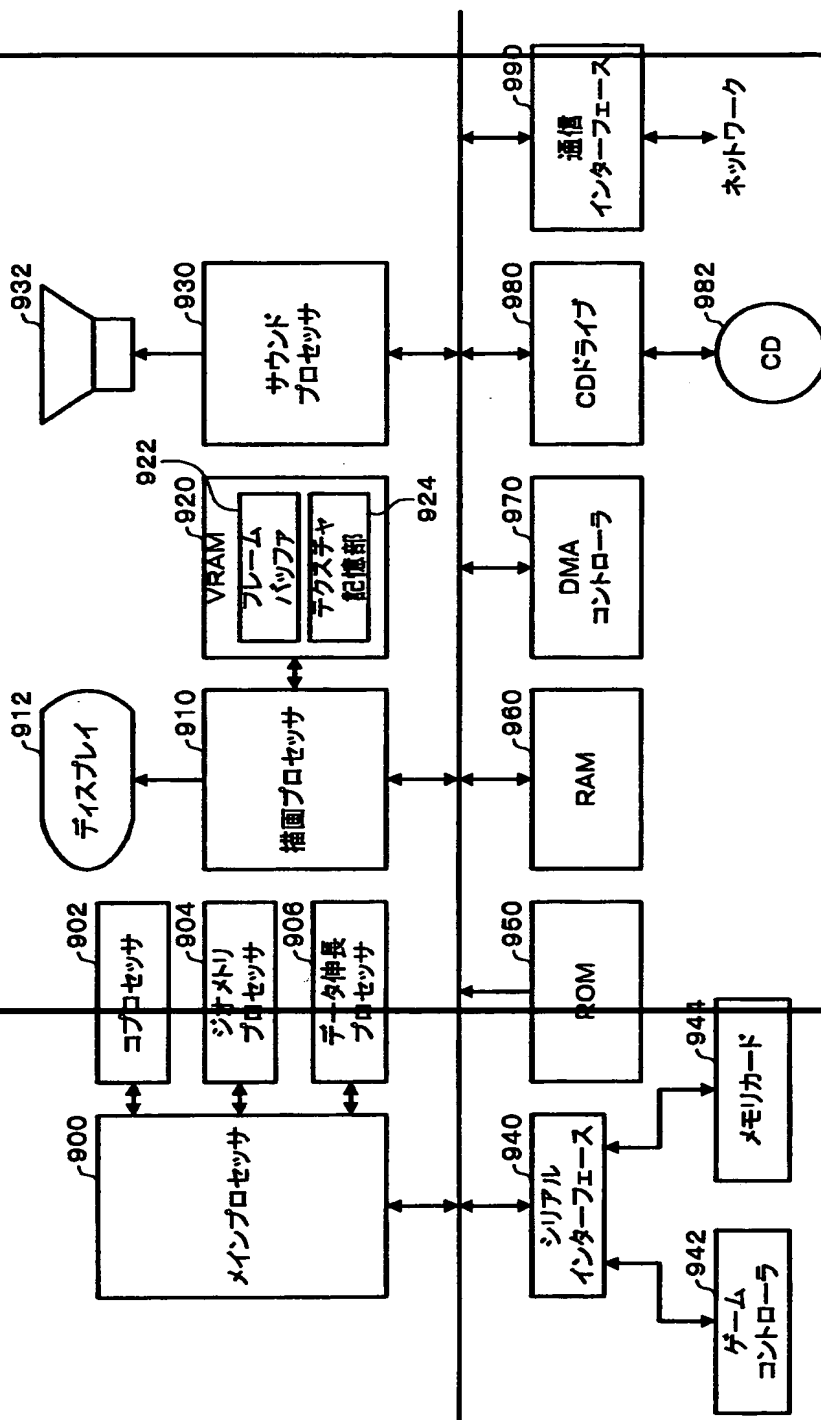
【図 2 7】



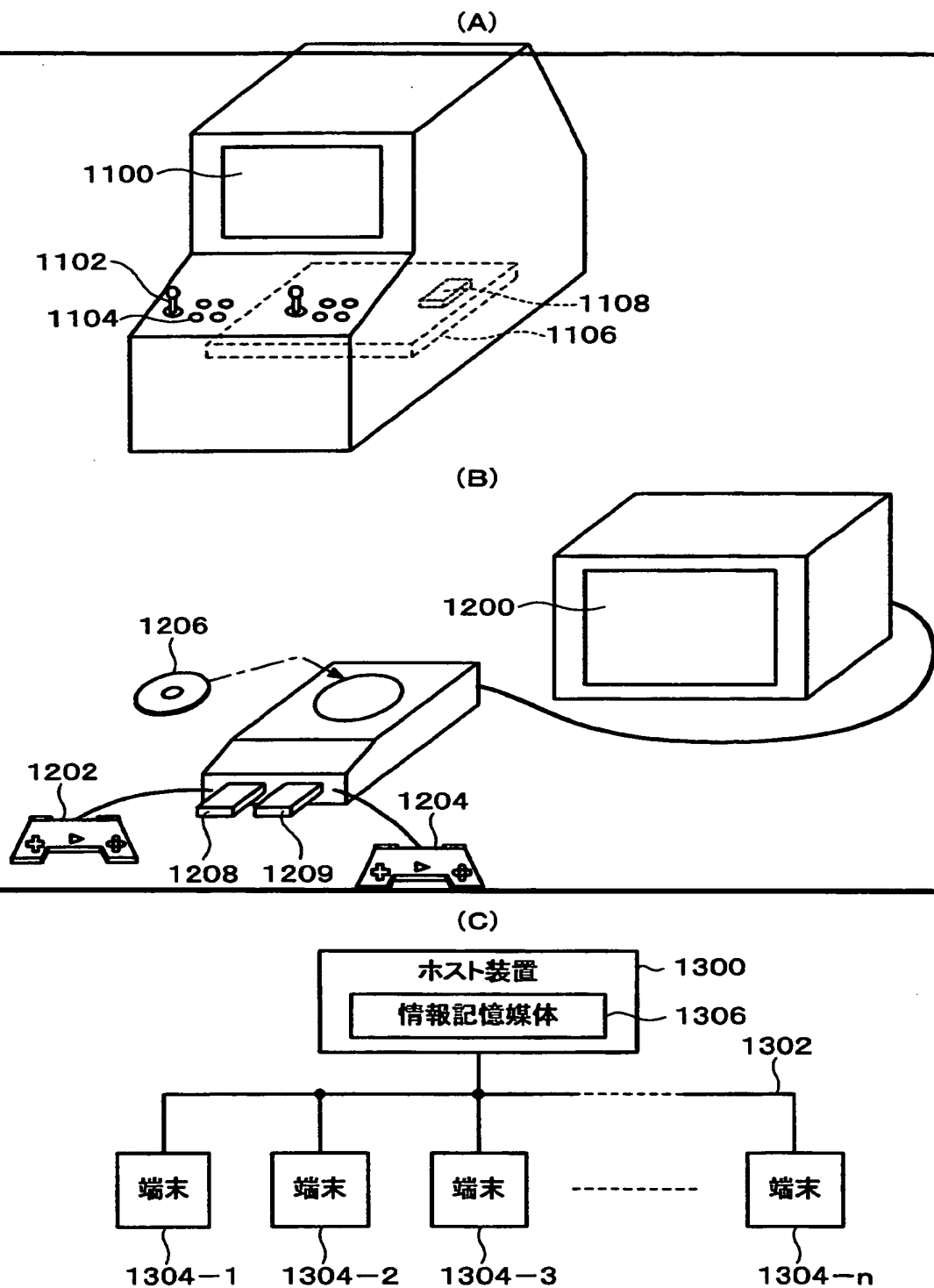
【図 2 8】



【図 29】



【図 30】



【書類名】 要約書

【要約】

---

【課題】 現実世界の視界画像のようにフォーカシングされた画像を、少ない処理負担で生成できるゲームシステム及び情報記憶媒体を提供すること。

【解決手段】 元画像の各画素のZ値を、インデックスカラー・テクスチャマッピング用のルックアップテーブルLUTのインデックス番号に設定して仮想オブジェクトに対するテクスチャマッピングを行い、元画像の各画素のZ値に応じた値に $\alpha$ 値を設定し、元画像とそのぼかし画像を合成する。Z値を、その最上位ビットよりも下位のビットI～Jにより構成されるZ2値に変換し、Z2値をLUTのインデックス番号に設定する。Z値のビットI～J以外のビットの値に応じてZ2値を所与の値にクランプする。Z値のビットM～N、ビットK～L ( $K \geq I \geq L > M \geq J \geq N$ ) をLUT1、LUT2のインデックス番号に設定してテクスチャマッピングを行い、Z値をZ3値、Z4値に変換し、Z3値、Z4値に基づきZ2値を求める。

【選択図】 図14



---

出 願 人 履 歴 情 報

識別番号

[000134855]

1. 変更年月日 1990年 8月23日  
[変更理由] 新規登録  
住 所 東京都大田区多摩川2丁目8番5号  
氏 名 株式会社ナムコ

**THIS PAGE BLANK (USPTO)**